Efficient Information Dissemination in Wireless Multi-hop.

Song Yean Cho

2008

EFFICIENT INFORMATION DISSEMINATION IN WIRELESS MULTI-HOP NETWORKS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF INFORMATIQUE
AND THE COMMITTEE ON GRADUATE STUDIES
OF ECOLE POLYTECHNIQUE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Song Yean Cho

September 22, 2008

Preface

Many protocols and applications in wireless multi-hop networks indispensably use broadcasting to delivery messages to all nodes in a network. Wireless routing protocols such as DSR, AODV and ODRMP broadcast route discovery messages, and many application protocols exchange query and response messages by broadcasting. Hence, the efficiency of broadcasting is critical to efficiency of these wireless protocols. Efficient wireless broadcasting is a topic that this thesis studies where efficiency is measured by the total transmission number to broadcast one unit data to the whole network. The study is done in two ways: extending classical methods and utilizing a novel method, network coding.

Classical broadcasting protocols are based on store-and-forward routing that views packets as atomic objects and lets a node store incoming packets in its local queue for some delay, before forwarding one or several copies of the packets to its neighbors. Among the classical broadcast protocols, the simplest and most widely used broadcast protocol is pure flooding. Pure flooding reliably provides total coverage of a network, but causes redundancy of packets, resulting in unnecessary collisions, and enormous inefficiency. To combat this inefficiency, many efficient broadcast protocols have been studied using probabilistic algorithms such as a gossip protocol, or algorithms based on topology control such as a Multi-Point Relay (MPR) based protocol. These broadcast protocols have been successfully used in many wireless protocols standardized in IEEE 802.11 and IETF.

One possible application of the broadcast protocols is wireless mesh networks standardized in 802.11 Working Group S. The 802.11 mesh networks face challenges such as inefficient broadcasting of so-called associated station information that is partially updated; there exists redundancy between newly updated data and already broadcasted data. This thesis addresses these challenges by introducing an Association Discovery Protocol (ADP) that is combined with MPR-based broadcasting and integrated with an extension of an Optimized Link State Routing (OLSR) protocol for the 802.11 mesh networks. The results of analysis

with modeling and simulation show that the protocol effectively decreases control overhead.

However, for continuous data traffic, there exists another possible broadcast solution, which could theoretically outperform the broadcast protocols with classical store-and-forward routing and would not require precise topology information. This solution is network coding. Theoretically, network coding enables an optimized solution for wireless broadcasting in a polynomial time, while it is NP-complete with classical store-and-forward routing and only approximation algorithms exist. Hence, broadcasting with network coding is, theoretically, at least as efficient as any other broadcasting.

This thesis focuses on utilizing network coding specifically as a practical solution for efficient wireless broadcasting. For the practical solution, this thesis proposes simple algorithms based on intuitive rationale about optimal efficiency of wireless network coding. The efficiency of the proposed simple algorithms is theoretically analyzed, and proven to be asymptotically optimal. The efficiency also is experimentally analyzed and shown, in some examples, to outperform other broadcasting with classical store-and-forward routing. Finally, from these simple algorithms we derive a practical broadcasting protocol executing with a simple and efficient coding method (that is, random linear coding). In addition, we propose a simple novel method for real-time decoding that could be combined with the practical network coding broadcasting protocol.

Acknowledgments

The thesis has been an inspiring, very challenging, but always interesting and exciting experience. I am deeply indebted to my advisor, Professor Philippe Jacquet and Dr. Cedric Adjih, for their support, encouragement, and invaluable advice during my Ph.D. study. Their wide knowledge and logical way of thinking have been of great value for me whenever I challenged issues of my study. Their understanding and personal guidance have provided a good basis for the present thesis.

I would like to thank Prof. Mario Gerla and Dr. Thierry Turletti for serving as my thesis readers and their invaluable suggestions. I would also like to Prof. Pierre Duhamel and Prof. Walid Dabbous for serving on my thesis committee and thank Prof. Francois Baccelli for serving as a chairman of my thesis committee.

I wish to thank ex-collaborators, Bang-Hoon Chun, a vice president, Kyoung-Hoon Yi, a principle manager in Samsung advanced institute of technology, and also thank Prof. Jin-Young Choi and Prof. Arthur H. Lee. Their advice has inspired and encouraged me.

I would like to thank my collaborators, Dr. Emmanuel Baccelli, Prof. Thomas H. Clausen and Prof. Pascale Minet whose suggestions have been very valuable.

I take this opportunity to express my profound gratitude to my friends, Yang-Jae, Hee-Young and Su-Jin for their moral support during my study.

Finally, this thesis is dedicated to my parents, who have given me continuous and unconditional support of all my undertakings, scholastic and otherwise. They always believe in me and never fail to provide all the support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Because a channel of wireless networks is unreliable and its devices have mobility, wireless networks are naturally dynamic; the dynamic features cause difficulty where all wireless protocols exchange messages. Hence, many wireless protocols use broadcasting as a way to delivery their messages. For example, routing protocols use broadcasting to distribute control messages and application protocols use it in a discovery phase.

The simplest way to execute broadcasting is pure flooding where every node forwards all initially-received packets. Pure flooding requires no global topology information and no control traffic, and reliably covers the whole network. However, pure flooding causes many redundant data receptions and unnecessary transmissions. These unnecessary retransmission wastes bandwidth which is deficient in wireless networks; it also wastes devices' energy which is limited in mobile devices. This inefficiency should be avoided, and hence this thesis studies efficient broadcasting. The efficiency that we study is energy-efficiency that is measured by the number of transmission to deliver one unit data. Hence, we can define an efficient broadcast protocol for this study as;
Efficient Broadcast Protocol : distributes a unit data (a packet) from a source to all nodes inside a network with the minimal transmission number.

For efficient broadcasting, a substantial number of protocols have been proposed; these broadcast protocols have aimed to decrease redundant data reception, while keeping the total coverage of a network. The protocols are usually based on either probabilistic algorithms [1–3], where packets are only forwarded with a certain probability, or Connected Dominating Set (CDS) algorithms [4–6], where packets are forwarded only by the limited number of forwarder nodes in CDS sets. We investigated the efficient broadcast protocols and extended one of them, an MPR(Multi-Point Relaying)-based protocol for wireless mesh networks.

The mesh networks face challenges such as inefficient broadcasting of a so-called associated station information, whose newly updated data has redundancy with already broadcasted data. To address these challenges, this thesis introduces an association discovery protocol (ADP) that is combined with MPR-based broadcasting and integrated with an extension of an OLSR protocol for mesh networks. The results of analysis with modeling and simulation shows that ADP effectively decreases control overhead and improves performance. The performance improvement is observed specifically for delay constrained control traffic; however, for continuous data traffic, a basic limitation may exist for broadcast protocols with classical store-and-forward routing.

The limitation comes from a classical communication model assuming one source and one destination such as a telephone system [7]. Unlike telephone networks from which the classical communication model originated, recent networks are shared networks where multiple nodes access the same channel such as Ethernet and wireless LAN. In the shared networks, there may exist several destinations from a source and multiple paths to several destinations, and the multiple paths would be organized by connecting several communication entities. Moreover, the multiple paths could be partially overlapped by having common links. With the classical communication model, all communication entities organizing the paths use classical store-and-forward routing, which treats packets as atomic objects and stores packets in a local buffer for some delay before forwarding them to other nodes. When store-and-forward routing is used, the capacity of the common links should be shared by multiple paths. This sharing would result in capacity shrinkage that is assigned to some destinations. As far as we keep the classical point-to-point communication model, it is hard to avoid the capacity shrinkage, regardless of a high compression rate at the source and reliable transmission without any error.

In order to overcome the basic limitation of the classical communication model, we may benefit from the existence of intermediate nodes in recent networks. In the shared networks, we can permit a new operation to intermediate nodes in addition to storing and forwarding of the classical communication model. The new operation is coding, and permitting coding at intermediate nodes is called as network coding.

Network coding enables each connection to use the whole capacity of the overlapped paths as if there exists only one connection in the network. In the seminal work of [8], Ahlswede et al. theoretically proved and qualified the benefit of network coding with multicast scenarios. The quantified benefit is a theoretical upper bound of network resource utilization (that is, the max-flow min-cut bound for multicasting) that classical store-and-forward routing may not achieve in some cases. This benefit is one motivation for this thesis to focus on network coding.

Another motivation is the existence of a simple and efficient coding method, random linear coding [9,10]. Random linear coding not only is simple but also guarantees to asymptotically achieve the theoretical upper bound of network resource utilization for multicasting [11]. When the simple random linear coding method is used, a remaining problem is how to minimize the transmission number to achieve the upper bound, or cost optimization.

The cost optimization of wireless broadcasting is NP-complete with classical store-and-forward routing [1]. But with network coding, we can obtain an optimal solution for wireless broadcasting in a polynomial time by solving a linear optimization problem [12, 13], and we can have a converging approximation in a decentralized way [13]. However, this thesis proposes a different and even simpler approach to obtain the optimal solution, rather than solving a linear optimization problem.

1.2 Contribution

The main contribution of this thesis follows: first, we extend an efficient broadcast protocol with store-and-forward routing for wireless mesh networks, propose ADP in order to improve broadcast performance, and analyze the protocol's performance with modeling and simulation. Second, we design simple and efficient broadcasting with network coding. We theoretically prove it to be asymptotically optimal in one dimensional torus networks and in two dimensional networks on Euclidean spaces, and in practice show that our broadcasting outperforms other broadcasting with store-and-forward routing.

We extend MPR-based broadcasting to wireless mesh networks with OLSR routing protocol by proposing an association discovery protocol (ADP) in order

to efficiently broadcast data whose content does not totally change and is partially updated. Moreover, we analyze the efficiency of the protocol with modeling and simulation. An MPR-based broadcast protocol is one efficient broadcast protocol that decreases the transmission number by reducing the number of forwarding nodes. We extend the MPR-based broadcast protocol in order to complement OLSR routing protocol for IEEE 802.11 wireless mesh networks. While taking advantage of MPR-based broadcasting, our extension introduces ADP in order to limit the overhead of repeating information in periodically broadcasted data [14]. This extension could support efficient broadcasting in mesh networks specifically when the data should be periodically broadcasted within a constrained delay, and the periodically broadcasted data would be partially updated each time. An example of such data is a database of mobile hosts in mesh networks, a so-called associated station information. The improved efficiency with our extension is controlled with our analytical model which models the database changes and control overhead to broadcast the changes. Our simulations on some representative graphs show efficiency improvement and accurate efficiency control with our modeling. This extension presents a successful solution based on efficient broadcasting with store-and-forward routing specifically for burst and delay constrained control traffic.

We introduce a simple and novel approach for efficient broadcasting with network coding that achieves the maximal-efficiency-benefit of wireless network coding. For continuous data traffic, there exists different approach that does not require precise topology information and theoretically outperforms other broadcast protocols with store-and-forward routing. This solution is network coding. Network coding creates *multiple benefits*: a coded packet transmission can simultaneously bring *innovative*, that is new and useful, data to several different receivers. When all receivers of a transmission can receive innovative data from the transmission, multiple benefits are maximized. In other words, in order to maximize benefits of wireless network coding, every transmitted packet must be innovative to all receivers. Formally,
maximal-efficiency-benefit of wireless network coding is achieved when every transmitted coded packet provides innovative data to all receivers.

Our basic approach to achieve the maximal-efficiency-benefit of wireless network coding comes from the following intuitive rationale and hypothesis. Each node needs to receive innovative data from every transmitted coded packet of each neighbor in order to achieve the maximal-efficiency-benefit of wireless network coding. If the necessity is satisfied, the received innovative data number per unit time at a node must be proportional to the expected number of neighbors (M). For the rationale and hypothesis, we follow simple reasoning (see details in section 4.4.2), which infers that every node could have the same transmission rate (*one*) if every node had the same expected number of neighbors (M) and the same receiving rate (a max flow, M) of innovative data. With this reasoning, we obtain the basic rate selection algorithm to assign *the Identical Rate to Other Nodes (IRON)* besides the source.

We theoretically prove asymptotical optimality of our simple broadcasting with network coding in large dense networks. We prove the optimal efficiency of our approach (IRON) in one dimensional torus networks in section 4.5.1 and also in two dimensional networks on Euclidean spaces in section 4.5.2. For the proof in two dimensional networks, we extend our basic approach IRON to IREN/IRON that assigns *Increased Rates for Exceptional Nodes and Identical rates for Other Nodes (IREN/IRON)*. The reason to introduce IREN/IRON is that two dimensional networks on Euclidean spaces have some border effects where nodes around the border definitely have fewer neighbors than the average. Hence, while most nodes nearly have the same number of neighbors (M), as networks become larger and denser, nodes around the border have exceptionally lower numbers of neighbors than others. Because of the border effects, IRON has to be extended for the two dimensional networks. With the extended approach (IREN/IRON), we prove that its maximum broadcast rate with network coding is the expected neighbor number (M), and from the proven maximum broadcast rate we deduce that our simple broadcasting with network coding is asymptotically optimal. The optimal efficiency means that every transmission is asymptotically beneficial and innovative to all receivers. This near-optimal efficiency could not be achieved by broadcasting with store-and-forward routing, and hence our simple broadcasting is, theoretically, at least as efficient as other broadcast protocols with store-and-forward routing. For one dimensional torus networks, the asymptotical optimality is proven with similar logic, except that IRON is used instead of IREN/IRON because of the non-existence of the border effects.

We develop static and dynamic heuristics for random networks and experimentally show through simulation results that our simple broadcasting based on our heuristics with network coding could outperform other broadcasting with store-and-forward routing. Our theoretically analyzed rate selection algorithms (IRON and IREN/IRON) assume that most nodes have the same number of neighbors

(M). However, in practical networks, the expected neighbor number is diverse, and there exist *the starving nodes* that have small numbers of expected neighbors ($\leq M$). This small neighborhood of a node causes that the node's receivng rate of innovative data is lower than the receiving rate of innovative data for maximal-efficiency-benefit of wireless network coding. Hence, we need a general rule to provide sufficient information to the starving nodes. To achieve this purpose, based on the neighborhood size our general rule statically determines which neighbors will starve, and lets a node *Increase its Rate for Most Starving Nodes(IRMS).* Although the static heuristic (IRMS) uses only the local topology, some experimental results show that more than 90 percent of nodes have the receiving rate of innotative data for maximal-efficiency-benefit of wireless network coding, M (a max flow and a capacity of min-cut). Hence the maximum broadcast rate could be closer to the transmission rate (M) for the maximal-efficiency-benefit of wireless network coding. Moreover, the cost to achieve the maximal-efficiency-benefit is low. As a result, some representative simulations showed that *the static heuristic, in practice, could outperform MPR-based broadcasting which uses precise two hop neighbor information, and could be as efficient as Connected Dominating Set (CDS) based broadcasting which uses a centralized algorithm with precise global topology.* Also we propose a dynamic adaptation of our simple broadcasting with network coding for spare networks. The static heuristic (IRMS) generally shows good efficiency in dense networks. However, we find that its efficiency decreases in some cases of small and sparse networks. We identify the cause: local static information is not sufficient to account for the irregularity of small sparse networks. Hence, we introduce the novel approach of using dynamic information beyond topology; the approach is not necessarily optimal, but it is efficient. This dynamic information is how much innovative data each node has at time t. Using the information, the dynamic heuristic technically increases a transmission rate of a node up to the largest gap of innovative data with its neighbors : Dynamic Rate Adaptation for Gap with Other nodes (DRAGON). The largest gap should be positive so that the node must have more innovative data than at least one neighbor, and otherwise the node must stop transmission until it has a positive gap with its neighbors. This dynamic adaptation of DRAGON behaves as a feedback control to equalize the innovative data gap in the same neighborhood. Some experimental results showed that in some sparse networks DRAGON could achieve efficiency levels of up to 70% of the theoretical optimal obtained by solving a linear optimization problem and 85% of the experimental optimal obtained by executing simulations with selected rates of the optimal solution. Moreover, some experimental results also showed that DRAGON could experimentally outperform the geometric approximate upperbound of a connected dominating set, which was obtained to consider the essentially overlapped space of two unit-disks for connection (see $E_{\mathrm{rel-cost}}^{(\mathrm{no-coding})}$ in section 4.3.2.2).

We present a practical protocol for efficient wireless broadcasting with network coding from the proposed heuristics in wireless mobile networks. In addition, we also propose a simple novel method for real-time decoding. As Codecast [15], CodeTorrent [16], MORE [17] and file sharing protocols for Wireless Mesh Networks [18] present practical protocols using a random linear coding method, we also present a practical protocol. While other protocols emphasize loss recovery and delay constraint, our protocol focuses on energy-efficiency which is achieved by our heuristics (IRON, IRMS and DRAGON). We detail operations of our protocol based on our algorithms with additional features : terminating protocol and real-time decoding. These operations include operations to transmit encoded data with random linear coding, and operations to exchange control data in order to organize the local information base. The local information base is required to decide when a node can stop transmitting encoded data while assuring eventual decoding at all nodes. In other words, our practical protocol guarantees eventual decoding with the restricted terminating condition; the condition demands an individual node to confirm that itself as well as its neighbors have sufficient data to recover all source packets. However, in practical wireless networks, neighborhoods may dynamically change and the changes could cause difficulty in maintaining the terminating condition for complete decoding at all nodes. Hence, for all nodes' eventual encoding, our practical protocol details the precise scheduling rules to transmit and stop encoded data with algorithms to manage the local information base used for the scheduling rules. Moreover, we also proposes a simple and novel method for real-time decoding for a practical solution. Generally, network coding does not guarantee decoding until a node receives all required data to recover all source packets. Such phenomena have been, in practice, frequently observed. In fact, it is theoretically known that the probability of decoding before the end of the whole network coding transmission, decreases as the efficiency of network coding increases [19]. Hence, efficient broadcasting with network coding could bring intolerable delay. Real-time decoding can decrease the intolerable delay by enabling a node to recover some source packets before the end of the network coding transmission. Some experimental results showed that our simple real-time decoding method improved the decoding rate of received data up to 80 percent.

1.3 Thesis outline

The contributions of this thesis are summarized as:
• Propose, analyze and evaluate an efficient broadcasting (ADP) which extends MPR-based broadcasting in order to efficiently broadcast data in wireless mesh networks with a hybrid structure.
• Introduce a simple and novel approach for efficient broadcasting based on network coding that achieves the maximal-efficiency-benefit of wireless network coding.
• Theoretically prove asymptotical optimality of our simple broadcasting based on network coding in large dense networks.
• Develop static and dynamic heuristics for random networks and experimentally show through simulation that our simple broadcasting based on our heuristics with network coding could outperform other broadcasting with store-and-forward routing.
• Present a practical protocol specification with a simple real-time decoding method for efficient wireless broadcasting with network coding from the proposed heuristics in wireless mobile networks.

This thesis is organized as follows: in chapter 2, we present an efficient broadcast protocol for wireless mesh networks. The broadcast protocol extends an MPR-based broadcast protocol and is combined with a newly proposed ADP. In chapter 3, we present a literature review in the context of a practical framework to use network coding, and then a literature review related with theoretical performance of multicasting with network coding. In addition, we also explain that rate selection is a key to designing an efficient broadcast protocol with random linear coding. In chapter 4, we propose, qualify and analyze the possibility of utilizing network coding as an alternative broadcasting solution for continuous data traffic. We first introduce our basic rate selection to achieve the maximal-efficiency-benefit of wireless network coding. Next, we theoretically study the efficiency of our basic rate selection and prove its efficiency to be asymptotically optimal in one dimensional torus networks and in two dimensional networks with extension. Then, we experimentally study its efficiency in practical networks and also proposed a dynamic heuristic DRAGON for irregular and sparse networks. In chapter 4, we present practical protocol specifications derived from our approach and algorithms. The specification details brief algorithms of data transmission with our algorithms using a random linear coding method, packet formats and a local information base. In addition, we also propose a simple and novel method for real-time decoding. Finally, in chapter 6, we summarize the thesis and present future work.

Chapter 2

Broadcast Protocols in Wireless Networks

In this chapter, we present efficient broadcast solutions with store-and-forward routing and one specific application of efficient broadcasting to mesh networks. As described in chapter 1, the efficiency that this thesis intends to achieve is energy-efficiency that is measured by the transmission number to deliver one unit data.

When efficiency is not considered, the simplest and most widely used approach to broadcast is pure flooding, where every node that receives a packet for the first time forwards it to its neighborhood. Pure flooding is simple and reliably provides a total coverage of a network. However, in terms of efficiency, pure flooding causes a lot of redundancy of packets, which results in unnecessary collisions and a huge waste of efficiency [1]. The inefficiency of pure flooding has caused many other efficient broadcast protocols to be studied. In this chapter, we review the efficient broadcast protocols with store-and-forwarding routing and investigate one efficient broadcast protocol, an MPR-based broadcast protocol for wireless mesh networks [14]. Moreover, we propose an additional protocol combined with an MPR based broadcast protocol in order to decrease control overhead to periodically broadcast data whose content is partially updated, and whose updated content has redundancy with already broadcasted data. That is an association discovery protocol (ADP). With modeling and simulation, we analyze the performance of executing ADP with MPR-based broadcasting.

2.1 Efficient Wireless Broadcasting with Store-and-Forward Routing

In this section, we categorize existing broadcast protocols with store-and-forward routing into four families: pure flooding, probability based approaches, area based approaches and neighbor-knowledge based approaches as is done in [20]. We explain briefly each category and details eight broadcast protocols belonging to four categories. Because simulation of [20] showed that an MPR-based broadcast protocol was generally efficient in various wireless network environments, we study the use of an MPR-based broadcast protocol in wireless mesh networks. In this section, we classify seven wireless broadcast protocols with store-and-forward routing into four categories as seen in [20]: Pure flooding, probability based approaches, area based approaches and neighbor-knowledge based approaches.

Pure flooding lets every node forward all initially-received packets. Probability based approaches use some basic understanding of the network topology to assign a probability for a node to forward. Area based approaches assume that nodes have a common transmission distance and let a node forward the received packet only if its forwarding reaches a sufficiently large additional coverage area. Neighbor knowledge approaches maintain neighborhood information by exchanging control messages such as a 'hello' message packet, and use the maintained neighborhood information to decide if a node forwards the packet or not.

◊ **Common Operations: Packet Processing and Forwarding**

In this section, we explain common operations for all protocols belong to four categories to process and to forward packets as seen in [6]. Each packet encapsulates one more broadcasted messages and the packets are embedded in UDP datagram for transmission over the network.

Messages can be flooded onto the entire network, or flooding can be limited to nodes within a diameter (in terms of number of hops) from the originator of the message. Thus transmitting a message to the neighborhood of a node is just a special case of flooding. When flooding any control message, duplicate retransmissions will be eliminated locally (i.e., each node maintains a duplicate set to prevent transmitting the same message twice).

Each message in a packet has its own message header, which includes an originator address field, a Time To Live field and a message sequence number field. An originator address field contains the address of the node that originally generated this message. When a node has several network interfaces and addresses, a node chooses one address as a main address and uses it as an originator address. But in this thesis, we assume that a node has a network interface and an address. A message sequence number field uniquely identifies each message generated from the originator node. Message sequence numbers are used to ensure that a given message is not retransmitted more than once by any node. A Time To Live field contains the maximum number of hops a message will be transmitted. Before a message is retransmitted, the Time To Live must be decremented by 1. When a node receives a message with a Time To Live equal to 0 or 1, the message must not be retransmitted under any circumstances. Normally, a node would not receive a message with a TTL of zero. Thus, by setting this field, the originator of a message can limit the flooding radius.

Upon receiving a packet, a node examines each of the message headers in a packet. A node may receive the same message several times. Thus, to avoid re-processing of some messages that were already received and processed, each node maintains a Duplicate Set. In this set, the node records information about the most recently received messages where duplicate processing of a message is to be avoided. For such a message, a node records a "Duplicate Tuple" (D-addr, D-seq-num, D-retransmitted, D-time), where D-addr is the originator address of the message, D-seq-num is the message sequence number of the message, D-retransmitted is a boolean indicating whether the message has been already retransmitted, D-time specifies the time at which a tuple expires and must be removed.

In a node, the set of Duplicate Tuples are denoted the "Duplicate set".

Thus, upon receiving a basic packet, a node must perform the following tasks for each encapsulated message:

1. If the packet contains no messages (i.e., the Packet Length is less than or equal to the size of the packet header), the packet must silently be discarded.

2. If the Time To Live of the message is less than or equal to zero), or if the message was sent by the receiving node, the message must silently be dropped.

3. Processing condition: if there exists a tuple in the duplicate set, where: (D-addr == Originator Address, AND D-seq-num == Message Sequence Number) then the message has already been completely processed and must not be processed again. Otherwise, the node processes the data in the message.

4. Forwarding condition: if there exists a tuple in the duplicate set, where: (D-addr ==

Originator Address, AND D-seq-num == Message Sequence Number), then the message has already been considered for forwarding and should be retransmitted again. Otherwise: the message must be considered for forwarding

"Default Forwarding Algorithm": the default forwarding algorithm is the following:

1. If the sender of the packet is not detected to be in the one hop neighborhood of the node, the forwarding algorithm must silently stop here (and the message must not be forwarded).

2. If there exists a tuple in the duplicate set where: (D-addr == Originator Address AND D-seq-num == Message Sequence Number). then the message will be further considered for forwarding if and only if: D-retransmitted is false.

3. Otherwise, if such an entry doesn't exist, the message is further considered for forwarding.

4. If the time to live of the message is greater than '1', the message MUST be retransmitted (as described later in steps 6 to 8).

5. If an entry in the duplicate set exists, with same originator Address, and same Message Sequence Number, the entry is updated as follows: D-time= current time + DUP-HOLD-TIME (duplication tuple hold time). D-retransmitted is set to true if and only if the message will be retransmitted according to step 4. Otherwise an entry in the duplicate set is recorded with: D-addr = Originator Address, D-seq-num = Message Sequence Number, D-time = current time + DUP-HOLD-TIME. D-retransmitted is set to true if and only if the message will be retransmitted according to step 4. If, and only if, according to step 4, the message must be retransmitted then:

6. The TTL of the message is reduced by one.

7. The hop-count of the message is increased by one

8. The message is broadcast on all interfaces (Notice: The remaining fields of the message header should be left unmodified.)

If after steps 3, the message is not considered for forwarding, then the processing of this section stops (i.e., steps 4 to 8 are ignored); otherwise, if it is still considered for forwarding then steps 4 to 8 are used.

◊ **Pure Flooding**

Pure flooding is initiated by a source node that broadcasts a packet to all its neighbors [21, 22]. Each neighbor of the source in turn forwards the packet exactly one time and this continues until all nodes inside the network receive the source packet. Pure flooding was proposed as a way to operate reliable broadcasting or multicasting in highly dynamic networks, for instance, as presented in [21]. In addition, Jetcheva et al proposed to use pure flooding in ad hoc networks with low node density and high mobility in [22]. Pure flooding is simple and reliably covers the whole network. However, pure flooding causes many redundant data receptions and unnecessary transmissions. Because of this inefficiency, a substantial number of more efficient protocols for wireless broadcasting have been proposed.

◊ **Probability based approaches**

The probability based approach in [1] is similar to pure flooding except that the nodes only forward received packets with a predetermined probability (rebroadcast-probability) that is less than one. In dense networks, multiple nodes share similar transmission coverage. Thus, although some randomly chosen nodes do not forward the received packets, all nodes can receive the broadcasted packets and the skipped forwarding can save network resources. In sparse networks, there exists much less shared coverage, and hence low rebroadcasting-probability prevents the broadcasted packets from reaching some nodes. When the rebroadcast-probability is one, the probability based approaches become identical to pure flooding.

Hass et al [3] also proposed a probability based approach, which is broadly called gossip. Gossiping also requires that each node forwards packets with a pre-specified rebroadcast-probability. The key idea of a gossip protocol is that correctly chosen rebroadcast-probability enables highly reliable broadcasting, even in sparse networks. However, choosing the correct value of rebroadcast-probability is a difficult problem. The correct value of rebroadcast-probability is closely associated with the topology of the network. Thus, without the total topology information, the value of rebroadcast-probability may not be correctly estimated. Moreover, the topology of wireless networks evolves over time due to transmission failures and link errors, and therefore, a suitably chosen gossip-probability may become sub-optimal over the network lifetime.

◊ **Area based approaches**

The purpose of area based approaches is to maximize the area size that a forwarded packet covers. Let us assume a node exists inside the source's transmission range r. If the receiving node is closely located to the source, forwarding of the receiving node can cover only a small additional area. On the other hand, if the receiving node is located at the boundary of the source's transmission range r, then the forwarded packet from the node can reach a significant additional area, precisely 60 percent [1]. Hence, the area based approaches allow a node to evaluate the size of additional coverage and then forward the received packet depending on the evaluation result. Note that the area based approaches do not consider whether nodes exist within the area or not, and only consider how large an area is covered by the forwarding.

To estimate precise additional coverage, a node needs to know its precise location and a transmitter's precise location. Global Positioning System (GPS) can be used [1] to get the precise location information. A node that finds its precise location using GPS, inserts its precise location in the forwarded packet. Then, any node that receives the packet can calculate the precise additional coverage. Only when the calculated additional coverage is larger than the threshold, the node forwards the packet.

◊ **Neighbor knowledge based approaches**

"Self pruning": is the simplest protocol of neighbor knowledge based approaches [23]. Self pruning uses precise one hop neighbor information that is obtained via a hello message. When a node forwards a packet, the node inserts its neighbor list in the forwarded packet. Then, any receiver of the packet compares its own neighbor list and the neighbor list in the received packet. If the comparison result indicates that retransmission from the node can reach any additional node, the receiver forwards the packet, and otherwise, does not forward it.

"Scalable broadcast algorithm (SBA)": uses two hop neighbor information organized inside each node instead of inserting its neighbor list in a forwarded packet [24]. To organize the two hop neighbor information, each node exchanges a periodic hello message including its identification (ID) and its neighbor list. Periodically exchanging the hello message enables a node to maintain the current two hop neighbor information. With the up-to-date two hop neighbor information, a node decides if it forwards or not. For example, node A broadcasts a packet and node B receives it. On receiving the packet, node B refers to its two hop neighbor information in order to find that node B has non-overlapped neighbors with neighbors of node A. If node B finds any non-overlapped neighbors, node B forwards the packet after adding the sender node to B's neighbor list. Although node B receives the duplicated packet from another neighbor, node B executes the same comparison process as it does for the packet from A. However, for forwarding decisions of the same packet within a predetermined transmission delay, node B forwards only once. The predetermined transmission delay is locally maintained inside a node and is randomly chosen from a uniform distribution between 0 and the highest possible delay interval.

"Dominant Pruning with Greedy Set Cover Algorithm": also uses two hop neighbor knowledge like SBA [23]. But, unlike SBA, dominant pruning requires each node to proactively choose some or all of its one hop neighbors as forwarders. Only these chosen nodes forward the broadcasted packet. In order to inform one hop neighbors of the selected forwarders, each broadcasted packet includes the selected forwarder list inside its forwarded packet header. On receiving the packet, a node checks if the packet includes itself in its forwarder list. If the node is included in the packet's forwarder list, the node forwards it. Before forwarding, the node decides which neighbors should forward the packet again using the Greedy Set Cover algorithm in [25]. The Greedy Set Cover algorithm excludes neighbors that already received the packet by recursively choosing one hop neighbors that cover the most two hop neighbors until all two hop neighbors are covered. The node's computed forwarder list is included in the forwarded packet from the node.

"Dominant Pruning with Extended Localized Algorithm": is another completely localized algorithm to construct CDS for dominant pruning and is proposed by Wu et al [26, 27]. Their algorithm also lets all nodes organize all of their two hop neighbors by exchanging their open neighborhood information with their one hop neighbors. With the two hop neighbor information, all nodes are initially unmarked as forwarding nodes. The process to choose forwarders uses the following simple rule: any node having two unconnected neighbors is marked as a forwarder. The set of marked nodes form a forwarding node set, with a lot of redundant nodes. Two pruning principles are provided to post-process the forwarder set S, based on the neighborhood subset coverage. A node u can be taken out from S, if there exists a node v with higher ID such that the closed neighbor set of u is a subset of the closed neighbor set of v. For the same reason, a node u will be deleted from S when two of its connected neighbors in S with higher IDs can cover all of u's neighbors. This pruning idea generates the following general rule [26]: a node u can be removed from S if there exist k connected neighbors with higher IDs in S that can cover all u's neighbors.

"Multi-Point Relaying (MPR)": also uses two hop neighbor information [5,6]. Each node selects a subset of neighbors called Multi-Point Relays as a forwarder set. This allows neighbors that are not in the MPR set to receive the message without forwarding it. To form the MPR set, each node continuously gathers precise two hop neighbor information using a hello message. Based on the precise two hop neighbor information, a node selects an MPR set that guarantees that all two hop away nodes get the forwarded packets, i.e., each two hop away node must be a neighbor of a node in the MPR set. The MPR set locally minimizes the duplicate retransmissions. (as seen in Figure 2.1, where MPR nodes are black dots).

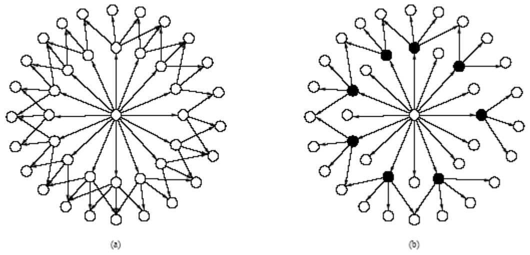


Figure 2.1: Pure flooding vs. MPR flooding

Because MPR sets are a kind of local dominating set, selecting minimal size MPR sets is also an NP-problem, but there exists an efficient heuristic which follows a "degree greedy" strategy that selects neighbors with the largest coverage of uncovered two hop neighborhoods [6]. The heuristic has good guaranteed approximation performance. The detailed algorithm to select an MPR set is described in algorithm 1.

Algorithm 1: MPR Selection Heuristic

```
1: Select as an MPR node all the neighbors of node $v$ that are the only neighbors of
   a two hop node from node $v$.
2: While there exist any uncovered two hop nodes from node $v$, node $v$ selects as
   MPR a neighbor of node $v$ that is a neighbor to the largest number of uncovered two
   hop nodes. If ties exist, select a node with the smallest node ID.
3: Discard any MPR node $u$ such that the MPR set excluding $u$ can still cover all
   two hop nodes
```

With the selected MPR sets using the heuristic in algorithm 1, the MPR-based broadcast protocol follows a simple rule to broadcast data: a node retransmits a broadcast packet if and only if the received packet is not received before, and the transmitter of the packet is an MPR. The main benefit of MPRs is that they allow for fewer forwarding nodes than in pure flooding, where all nodes are forwarding nodes as seen in Figure 2.1. We use the efficient MPR-based broadcast protocol for mesh networks, which is used in an early draft of the Institute of Electrical and Electronics Engineers (IEEE) 802.11 Working Group S.

2.2 Efficient Broadcasting Extension to Wireless Mesh Networks

MPR-based broadcasting is an efficient broadcasting method with classical store-and-forward broadcasting in wireless networks. MPR-based broadcasting reduces the transmission number by decreasing the size of a forwarder set. One practical application is to use the efficient MPR-based broadcast protocol with an OLSR protocol for mesh networks, which is specifically standardized in IEEE 802.11 Working Group S. When we use the two protocols, an additional protocol should be added. The additional protocol is ADP, which decreases the control overhead distributed using an MPR-based broadcast protocol. We explain the extension of an MPR-based broadcast protocol in wireless mesh networks as follows: we first explain wireless mesh networks in terms of their structure and contrast with other wireless networks such as Mobile Ad hoc Networks (MANET); we then describe other protocols applied to wireless mesh networks; we explain an OLSR protocol using MPR-based broadcast protocol to distribute control messages;we introduce an association discovery protocol to decrease control overhead; we analyze the performance by using the association discovery protocol with a model; we detail experimental results and summarize the results to apply MPR-based broadcasting with an OLSR protocol and an association discovery protocol.

◊ **Introduction of Wireless Mesh Networks**

Wireless Mesh Networks (WMNs) have been proposed as a prominent network solution for ubiquitous networks [28], and industry standard groups such as IEEE 802.11 [29–31], 802.15 [32] and 802.16 [33] are developing new specifications for WMNs. The recent broad interest in WMNs is due to two major features of WMNs. First, WMNs are capitalizing on the self-organizing aspects of ad-hoc networks, and the ease of deployment with minimal preparation, requiring less administration and maintenance. Second, WMNs (particularly, for 802.11 networks) can capitalize on a broad installed base, and on a technology that has been already developed: they enable mesh clients to access networks, without requiring them to execute an additional routing protocol (for instance, 802.11 stations). Instead of running a complex routing protocol, clients simply associate themselves with mesh routers and rely on mesh routers to access the whole WMN; the routing on the wireless mesh routers is transparent for the client. This feature may also be used by design, in order to isolate the routing functionality from the clients; then different deployment of WMNs with different features, could still accommodate the same wireless mesh client. This also allows lighter clients and simpler participation in heterogeneous networks.

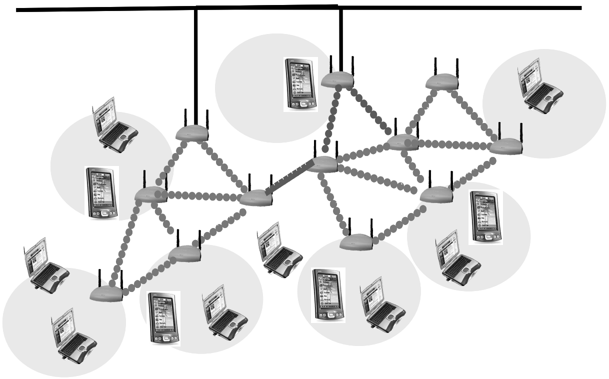


Figure 2.2: Architecture of hybrid wireless mesh networks

Figure 2.2 illustrates two main components of WMNs: *mesh routers* and *mesh clients*. Mesh routers create multi-hop networks that are self-configuring and self-healing, by executing routing protocols. The routing protocols for WMNs need not be much different from those of wireless ad hoc networks in general and Mobile Ad hoc NETworks (MANET) [34] in particular. For MANETs, several routing protocols such as OLSR (Optimized Link State Routing Protocol) [6], AODV (Ad hoc On demand Distance Vector) [35], TBRPF (Topology Broadcast based on Reverse-Path Forwarding) [36] and DSR (Dynamic Source Routing) [37] have been proposed, and some variants of such protocols are being used for WMNs.

Unlike mesh routers, mesh clients should associate with mesh routers to access networks. To associate with mesh routers, mesh clients use specific attachment protocols such as page/inquiry for Bluetooth or *association* request/reply for IEEE 802.11.

Obviously, in order to accommodate the mesh clients participating in a WMN, the mesh routers should perform new operations in addition to the operations that the classical ad-hoc network routers perform. The basic MANET routing protocols [6, 35–37] are indeed operating on flat topology; hence it is insufficient to find paths between mesh clients of the hybrid network. In order to collect complementary information, mesh routers can execute an additional auxiliary protocol.

In this chapter, we propose such a solution: an association discovery protocol (ADP) for mesh networks. Using ADP, mesh routers can compute routes to the mesh clients associated with other mesh routers (by means of an association table). Our protocol is simple and general; it can be used to complement any ad-hoc network routing protocol with any efficient broadcast protocol. However, it targets more specifically 802.11s, for which it was initially developed. Additionally, because it is proactive in nature, it is more naturally suited to run with MPR-based broadcast protocol and the optional 802.11s OLSR protocol's variant, Radio-Aware OLSR (*RA-OLSR*), integrated in an early draft of 802.11s [38].

◊ **Routing with Store-and-Forward Broadcasting in Wireless Mesh Networks**
A hybrid network structure is not a unique feature of WMNs. Hybrid routing protocols such as FSR [39], ZRP [40], CBRP [41], or Landmark (Landmark+OLSR [42]) have been explored for MANETs with some hybrid structures. But their hybrid structure is in general different from that of WMNs in two aspects: WMNs admit the presence of non-router nodes and they can use a flat address space.

Assuming the existence of a routing protocol between mesh routers, the problem that we are trying to solve is straightforward: *how can we exchange information indicating which station (mesh client) is attached to which mesh router ?*

Routing protocols themselves integrate some functionalities to inject some external routes in the network. For instance, OSPF has its own mechanisms to import non-OSPF routes with specific types of OSPF LSA messages [43, 44]. These messages are used to exchange routes that are external to the OSPF network, and are then used to complement the routes computed by OSPF.

Similarly, in order to integrate nodes without routing functions, OLSR also provides a mechanism called *Host and Network Association* (*HNA*) to exchange information about routes to non-OLSR nodes. However, in the current specification [6], HNA in [6] is not well adapted to non-OLSR nodes for which the last hop (an OLSR node) may be changing quickly; an OLSR node periodically distributes HNA messages listing all its non-OLSR nodes with MPR-based broadcast protocol. There are two problems. The first one is the necessity of performing a full diffusion of the HNA-database when a node disappears, and the second one is the large overhead of such messages, when the number of associated stations is large. An approach based on both layer two and layer three could improve the efficiency on HNA messages [45].

Looking at a broader picture, more general protocols also exist to diffuse distributed sets of information, such as OSPF-style database exchange protocols for OLSR [46], Gossip-based protocols [47] and reactive protocols using on-demand request/response. Unlike these protocols, our protocol is specifically designed to address the following information exchange problems:

- Information items (which mesh client is associated with which mesh router) with sequence numbers

- Medium-sized information set (potentially too large to be diffused periodically, but for which a single level of hierarchy is sufficient)

The main feature of the ADP proposed is to limit the overhead of repeating information that is already broadcasted and partially updated. The ADP is combined with MPR-based broadcasting to take advantage of the optimized flooding mechanism. In the next section, we explain OLSR briefly.

◊ OLSR Routing Protocol with MPR Broadcasting

An OLSR protocol, which can be used to select paths in WMNs, is a proactive link-state routing protocol, employing periodic message exchanges to update topological information in each mesh router. Because an OLSR protocol is specifically designed to operate in the context of wireless multi-hop networks such as MANET, it provides an optimized flooding mechanism, called Multi Point Relay (MPR)-based broadcasting, which is used to diffuse topology information. MPR based broadcasting optimizes flooding by minimizing the redundant retransmissions of Topology Control (TC) messages. Minimization is achieved by limiting the forwarders of messages to some MPRs. A set of MPRs relays is a small set of neighbors through which a sender can reach all two hop neighbors.

As Figure 2.1 shows, messages can be broadcast from the source to the entire WMN through MPRs' relaying. The first figure in Figure 2.1 shows the retransmission of pure flooding, where every node retransmits the packet and the second figure shows the retransmission of MPR based broadcasting, where only MPR nodes retransmit the packet. Only with MPRs' retransmission, can the source successfully broadcast the packet to every node in the network. In addition, a node can reach any node in the network only utilizing links between MPRs. Thus setting up the paths may not require all links between mesh routers. Information about links with MPRs is sufficient, by property of the MPR (and because all nodes have selected an MPR set). Thus mesh routers can simply transmit the addresses of all their MPR selectors with their addresses in TC messages. Even if the topology information obtained from received TC messages is a partial topology of the whole WMN, the shortest path obtained from this partial topology has the same length as the shortest path from the full topology [48].

For the specific application of 802.11s Mesh Networks, a variant of OLSR is used as an option: *Radio-Aware OLSR*. It integrates essential features for selecting links with good quality.

Now, considering the hybrid structure of WMNs, the association between mesh clients and mesh routers must be discovered in order to complement the topology information of the mesh routers.

When delivering messages, One or more messages are encapsulated in each packet and the messages share a common header. In addition, none of these messages requires that a node forwards the same messages more than one time. In other words, the duplicity of received messages is checked and a node only forwards the messages that are received for the first time.

◊ Association Discovery Protocol with OLSR Routing and MPR Broadcasting

ADP, which is combined with MPR-based broadcast protocol, can be used to complement any routing protocol, as previously indicated. However, we designed it specifically with 802.11s Mesh Networks in mind: "802.11s" is the task group 'S' of the 802.11 working group, in charge of proposing mesh networks extensions to 802.11. Indeed, our protocol had been integrated in the early drafts of 802.11s in [38] (*"Associated Station Discovery"*).

Such mesh networks distinguish two types of nodes[1]:

[1] *Mesh Points*, which are clients able to run the routing protocol are also possible in 802.11s, but for our description, they are identical to mesh routers without clients

- Mesh routers, which run the routing protocol, and are able to associate mesh clients.

- Mesh clients (legacy 802.11 STA), which *associate* themselves with mesh routers.

The base mechanism of ADP is the following: mesh routers diffuse the whole set of mesh clients associated with themselves using MPR-based broadcast protocol. It is hence a proactive protocol, similar in spirit to the functioning of an OLSR protocol; in both cases the topology (associated station) information messages must be refreshed within a guaranteed interval (i.e. before the expiration). However, in addition to periodic message exchange, in case of topology/association change, faster updates are possible, and are desirable, in optimized implementations. As explained in section 2.2, mesh clients rely on mesh routers to compute paths to their destinations, and these computations require both routing tables and association tables. To obtain the two types of database (tables), mesh routers exchange the list of mesh clients associated with themselves in addition to the traditional information from the routing protocol. They store and maintain the received information using two information bases:

- *Local Association Base (LAB):* Each mesh router keeps track of mesh clients associated with itself. This information base is updated whenever associated mesh clients leave or new mesh clients join.

- *Global Association Base (GAB):* Each mesh router maintains a GAB to record which station is associated with which mesh router in the entire WMN. Upon receiving LAB from all other mesh routers, the GAP is updated. Hence the GAB contains the union of all the LAB, of each mesh router in the WMN.

To fill the GAB of other routers, each mesh router broadcasts its full LAB periodically. These additional periodical messages generate overhead. This feature brings an interesting trade-off between bandwidth use and reactivity to changes. Considering that a LAB is not modified when there is no new association or disassociation, an optimization is possible: diffusing only updates (newly associated, or disassociated stations).

There is, however, a fundamental issue with the exchange of *differential* updates: message loss. When a differential update is lost, it is not sufficient to detect the loss of the message and have it repeated because additional changes could occur (including re-association of disassociated stations for instance) before the message repeats. It is complex to provide recovery after message loss, using uniquely differential updates, without maintaining a time-ordered log of all changes. Hence, an additional mechanism to check information base *consistency*, and a mechanism to exchange full (non-differential) updates has to be integrated.

A main feature of our protocol, is that it restricts itself to the two last mechanisms, while offering performance comparable to differential updates. To do so, every mesh router with a large number of mesh clients, can divide its LAB into several smaller *blocks*. Then updates of association/disassociation of one or a few stations, are made by diffusion of the one or the few small blocks containing those addresses. Since in 802.11 type of networks, the MAC overhead is important for small packets, the cost of either sending one address or sending a few addresses in the same message is expected to be on the same order of magnitude.

"General Protocol Operation of association discovery protocol" : Precisely, the protocol operates with three kinds of messages.

- `LABA` messages: used for diffusion of the *full* of content of one block, some blocks, or all blocks of the LAB of the originator node.

- `LABCA` messages: used for diffusion of *checksum* of one block, some blocks or all blocks of the LAB of the originator node. The purpose of such messages is to avoid repetition of already widely disseminated information: a checksum of the block is diffused instead of the block itself.

- `ABBR` messages: used to require a set of blocks, typically upon receiving a mismatching `LABCA` checksum for some block(s) owing to lost LABA messages.

Notice the inherent tradeoff on block size: larger blocks imply larger LABA messages, and small blocks imply larger LABCA messages.

With this set of three messages, a large number of protocol variants could be implemented, depending on client mobility, packet loss, and communication patterns. Generally, blocks, or messages intervals, may all be adjusted dynamically, and need not be identical for all blocks. However, for interpretability the size and the interval would need to be predetermined. In addition, all messages need not be sent by/to the mesh router originally holding LAB information. And LAB updates may be sent preferentially to some more critical mesh routers.

Note also that instead of a checksum, a simple sequence number per block could alternatively be used, matching the methods of OSPF for instance. The advantage of the checksum is that it allows to arbitrarily change the repartition of the addresses into blocks, a feature that is not currently used.

In the following, we will focus on a simple variant of the protocol described in the following section.

"Simple Protocol Operations of Association Discovery Protocol": In this section, we are describing simple protocol operations, which still uses the ability to decrease the ADP overhead using checksums.

In these protocol operations, there are two operating modes: a *full* mode and a *checksum* mode. Initially mesh routers operate in a full mode. After a given duration, mesh routers that did not sense any change in their LAB, will start to operate in a checksum mode. Any change in their LAB resets mesh routers back to operate in a full mode.

- Full Mode: In full mode, mesh routers periodically broadcast the whole content of their LAB (all blocks).

- Checksum Mode: If the network becomes stable and few mesh clients are moving, the LAB of mesh routers will not change much. In this case, mesh routers avoid generating heavy overhead by simply diffusing checksums representing the status of their LAB rather than sending all their content, in `LABCA` messages.

Upon receiving these checksums, a receiving mesh router cannot populate entries in its GAB, but it can verify them. If a checksum mismatch is found, this implies that the mesh router has missed some updates of the LAB of the mesh router that originated the message. In this case, the mesh router will send a request (`ABBR`) for the mismatching parts to the original mesh router in order to restore the consistency of its GAB by recovering the lost information. Upon receiving these requests, the mesh router switches to a full mode, thus emitting the contents of its whole LAB.

◊ Analytical Model of Association Discovery Protocol with MPR Broadcasting

As described in the previous section, the mesh routers executing the ADP are switching between two operation modes, depending on the stability of the associated mesh clients.

Messages are emitted periodically, and after a fixed number of LABA emission intervals without message loss (i.e. without ABBR requests) and without changes in station association (i.e. change in the LAB), the mesh routers switch to a checksum mode. The fixed number of LABA emission intervals is a parameter of each mesh router in the system. In order to evaluate the performance of the system, the following two metrics are used: *Packet Error Rate (PER)* (i.e. number of packets lost due to obsolete routing information) and the control overhead. In this section, we provide an analytical model of the performance for this protocol.

"Packet Error Rate (PER)": In our model, mesh clients move independently and their movements are not synchronized. To focus on PERs' change according to the ADP's operations, data packets are assumed to be lost only when mesh clients move from the sojourned mesh router, and notifications of this movement are lost. The loss time when the position of mesh clients is not known is shorter than the sojourn time when mesh clients associate with the mesh router.

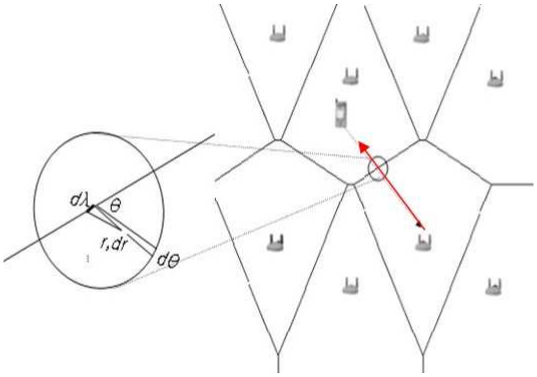


Figure 2.3: The change of association with mesh clients' mobility

Under this assumption, mesh router i changes its operation mode from full mode to checksum mode when its LAB does not change for h_i x t_i, where h_i is mesh router i's control message distribution interval and t_i is the number of message intervals for mesh router i to switch from full to checksum mode. As mesh clients move, they associate with each mesh router several times for some duration. We call *lifetime* (L_i^α) the duration when a mesh client α associates with a mesh router i, and we call *frequency* (τ_i^α) the number of times that one mesh client α sojourns with mesh router i. Packets are lost as long as the control message is not received (*the control information loss duration*). PER between mesh clients can be derived from the *average control information loss duration (ACILD)* between mesh routers. Once the ACILD is known, considering all cases where a mesh client α can be located and utilizing the fact that a mesh client β stays for an average time L_j^β, the PER from mesh client α to mesh client β is:

$$\sum_{(i,j),i\neq j} \tau_i^\alpha L_i^\alpha ACILD_{ij} \tau_j^\beta \tag{2.1}$$

In addition, because mesh clients move with a random walk in our model, τ_i^α and L_i^α depend on the area that mesh routers cover. More precisely, $\tau_i^\alpha L_i^\alpha$ is equal to $\frac{A_i}{A}$, where the total network area is A and mesh router i covers an area A_i. Similarly, as mesh router i has longer border, the probability that mesh client α crosses the border becomes higher, and the frequency with which mesh client α sojourns with mesh router i (τ_i^α) also increases. Precisely, τ_i^α is proportional to B_i, the border of the area A_i covered by mesh router i. When the mesh client α moves at speed v^α, the probability that it crosses the border on the portion $d\lambda$ during time interval $(t, t + d\lambda)$ is:

$$\frac{d\lambda sin\theta}{2\pi Ar} rdr \tag{2.2}$$

We can derive Equation 2.2 from the mesh router's area as seen in Figure 2.3. At time t, the mesh client α must be in the half of the disk of radius $v^\alpha dt$ and moves toward the portion $d\lambda$ as seen in Figure 2.3. Considering the mesh client α at a distance between r and $r + dr$ in the cone at angle θ, the probability that the mesh client α is within this area is $\frac{rdrd\theta}{A}$ when the angle at which mesh client α sees the portion $d\lambda$ is $\frac{d\lambda sin\theta}{r}$ and the speed of mesh client α toward this direction is $\frac{d\lambda sin\theta}{2\pi r}$. Therefore, the probability that the mesh client α crosses the border on the portion $d\lambda$ during the time dt is:

$$\int_0^\pi d\theta \int_0^{vdt} \frac{d\lambda sin\theta}{2\pi Ar} rdr = \frac{v^\alpha d\lambda}{\pi A} \tag{2.3}$$

On the total border length B_i, the probability that the mesh client α crosses B_i becomes $\frac{v^\alpha \times B_i}{\pi A}$. Consequently, packet error rate between the mesh client α and β, $PER_{\alpha\beta}$ is equal to

$$\sum_{(i,j)i\neq j} \frac{v^\alpha \times B_i}{\pi A} L_i^\alpha \times ACILD \times \frac{v^\beta \times B_i}{\pi A} \tag{2.4}$$

To compute the PER using Equation 2.4, we need the ACILD, which can be derived from r_ij, the probability of losing control messages between mesh router i and mesh router

j. When a mesh router does not operate the ADP, it always runs in a full mode, and then ACILD is

$$h_i \sum_{k=0}^{infty} kr^k(1-r) = \frac{h_i}{1-r_{ij}} \tag{2.5}$$

Unlike for Equation 2.5, mesh routers may switch between operation modes, and we have to consider the control information loss duration for both modes. Mesh router i fails to switch its operating mode to a checksum mode when it does not receive any control messages for h_i intervals. Thus, the probability that mesh router i fails to switch its operation mode to a checksum mode for duration h_i is $r_{ij}^{h_i}$, and the probability that mesh router i succeeds is $1 - r_{ij}^{h_i}$. Accordingly, when mesh router i fails to switch to checksum mode, its control loss duration is:

$$t_i + \frac{1}{1-r_{ij}} \tag{2.6}$$

In addition, when mesh router i can switch to a checksum mode, its control information loss duration ($CILD_{succeed}$) becomes:

$$\frac{t_{ij}}{1-r_{ij}} \times \sum_{k=0}^{h_{i-1}} kr_{ij}^k(1-r_{ij}) = \frac{1-r_{ij}^h}{1-r_{ij}} - hr_{ij} \tag{2.7}$$

Because mesh routers running the ADP must consider loss duration both ways, average control loss duration (ACILD) is:

$$(1-r_{ij}^{h_i} \times (Equation(\ 2.6))) + r_{ij}^{h_i} \times ((Equation\ (2.7)) + ACILD). \tag{2.8}$$

Reducing Equation 2.8, ACILD is $\frac{r_{ij}^t}{1-r_{ij}{}^t}(t_i + \frac{1}{1-r_{ij}}) + \frac{1-r_{ij}^h}{1-r_{ij}} - hr_{ij}$

Finally, the difference of a control loss interval between a mesh router with ADP and without ADP becomes $h_j \frac{r^t}{(1-r^t)(1-r)}$ and the difference of PERs between a mesh router with ADP and without ADP becomes

$$\sum_{(i,j)i \neq j} \frac{v^\alpha \times B_i}{\pi A} L_i^\alpha \times h_j \frac{r^t}{(1-r^t)(1-r)} \times \frac{v^\beta \times B_i}{\pi A} \tag{2.9}$$

To optimize the ADP, mesh routers choose t_i minimizing expression 2.9 under given conditions.

”Control Overhead” : When mesh router i operates in a full mode, the size of the control messages for distributing its LAB is proportional to the average number of mesh clients associated with mesh router i. In contrast, the overhead in a checksum mode does not change regardless of the number of clients assuming the number of blocks is fixed. If the ADP represents each mesh client in C_1 bytes and the hash function generates checksum output as C_2 bytes where $P^i_{checksum}$ is the probability that mesh router operates in a checksum mode, the decreased overhead per h_i by introducing checksum mode is

$$C_1 M_i - C_1 P^i_{full} M_i + C_2 P^i_{checksum} = P^i_{checksum}(C_1 M_i - C_2) \tag{2.10}$$

As Equation 2.10 shows, the overhead for mesh router starts to decrease if M_i is greater than $\frac{C_1}{C_2}$. If $C_1 = C_2$, the control overhead decreases when the mesh clients just outnumber the mesh routers. This overhead decrease is also related to the mesh client's mobility, L^α_i and τ^α_j. As mesh client α sojourns with mesh router i for L^α_i with τ^α_j frequency, the average number of mesh clients associated with mesh router i (M_i) equals $\sum_\alpha \tau^\alpha_j L^\alpha_j$. Combining this sum with Equation 2.10, the overhead decrease is:

$$P^i_{checksum}(C_1 \sum_\alpha \tau^\alpha_j L^\alpha_j - C_2) \tag{2.11}$$

To use Equation 2.11 we should find the $P^i_{checksum}$, the probability that the mesh routers are in checksum mode. $P^i_{checksum}$ depends on when mesh routers change their operation mode from full to checksum. This is when their LABs do not change for h_i x t_i. It means that the possibility changes according to the mesh client's mobility, i.e., lifetime (L^α_i) and frequency (τ^α_i). If mesh clients move from the area of one mesh router to the area of another mesh router with Poisson distribution, the possibility that the LAB of mesh router i does not change for (h_i) is:

$$e^{-(2\frac{M_i}{L_i})h_i} \tag{2.12}$$

In Equation 2.12, M_i is the number of mesh clients associated with mesh router i. M_i is then related with the mobility of mesh clients and their number. More precisely, M_i is $N \times \tau_i L_i$, where N is the total number of mesh clients. Similarly to τ^α_i and L^α_i, τ_i is the sum of the frequency with which mesh clients associate with mesh router i, and L_i is the sum of the sojourn time of all mesh clients with mesh router i. As a result, the probability that mesh router i operates in checksum mode $P^i_{checksum}$ is:

$$e^{-(2N\times\tau_i\times h_i)\times t_i} \tag{2.13}$$

Finally, we can choose t_i to adjust the control overhead decrease using the following equation:

$$e^{-(2N\times\tau_i\times h_i)\times t_i}(C_1\sum_{\alpha}\tau_j^{\alpha}L_j^{\alpha}-C_2) \tag{2.14}$$

◊ **Experimental Results of Association Discovery Protocol with MPR Broadcasting**

To evaluate our analytical model and the ADP itself, we performed simulations using grid topologies as shown in Figure 2.4.

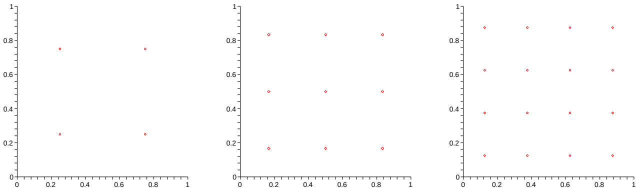


Figure 2.4: Mesh routers in grid topology

We compared the PER and control overhead, in two sets of simulations: the first ones only with full mode, the second with both full and checksum modes. Simulations were run with the parameters given in Table 2.1, (except for the switching interval). To remove other factors affecting PER and control overhead, a Null MAC was used (no collision). The mobility is the maximum distance where one 802.11 legacy station moves one time and is proportional to the transmission range.

In Figure 2.5, X axis presents the size of movement when the network is a 1 x 1 field network, and Y axis presents lost packet number per one packet delivery. If mesh routers do not use the ADP, they always distribute the whole LAB. Though distributing the whole LAB can improve PER, it consumes considerable bandwidth. To measure the amount of PER decrease by operating the ADP, we compare PERs from a mesh router with both checksum and full modes with PERs from a mesh router distributing the whole LAB. Figure 2.5 shows PERs from these two kinds of mesh routers. The PERs from a mesh router using ADP is slightly lower than the other, but their difference is just less than 0.002. This result

Table 2.1: Simulation and Scenario Parameters

Simulation Parameters	
Control msg loss rate per link	0.1
Mobile station num	3 x Mesh AP num
Control message interval	2 sec
Transmission range between Mesh routers	0.53(2x2), 0.35(3x3), 0.26(4x4)
Scenario Parameters	
802.11 legacy stations speed	0.01,0.03,0.05 x TR
Simulation time	20000 sec
Mobility model	Random walk
Pause time	5 sec
Traffic pattern	constant bit rate, 512 bps
Max connection	0.1 x mesh client num

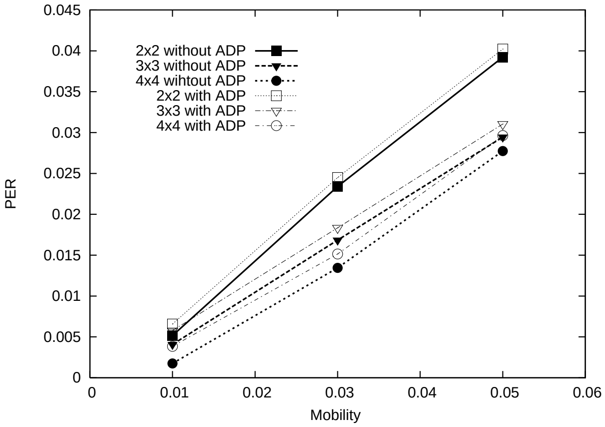


Figure 2.5: Comparison of PERs with and without ADP optimization

shows that ADP allows the mesh router to maintain its PERs at the same level as when it distributed the full LAB periodically.

In Figure 2.6, X axis presents the size of movement when the network is a 1 x1 field network, and Y axis presents decreased byte per one packet delivery.

While keeping its performance as described above, ADP decreases the control overhead as shown in Figure 2.6 after adjusting the control message distribution interval h_i. If a mesh client moves slower than 0.05 x transmission range per second, for example at 0.003 x transmission range, the control overhead decreases from 8 to 2 bytes per control message distribution interval h_i. The fact that the mesh router can decrease the control overhead using ADP supports the hypothesis that ADP saves bandwidth by optimizing the control

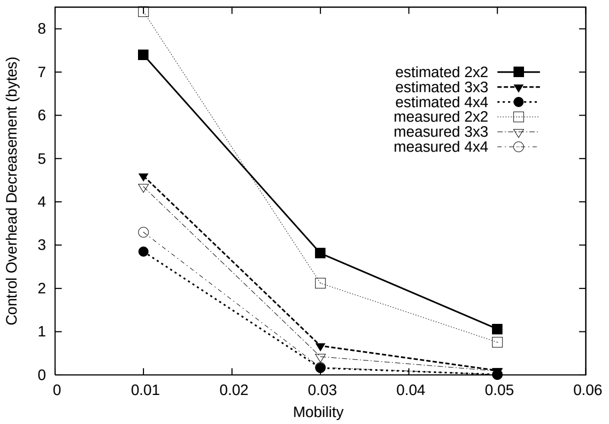


Figure 2.6: Control overhead decrease

messages to distribute mesh routers' LAB.

In Figure 2.6, control overhead decreases to almost zero with 0.05 mobility in grid 4x4. This overhead fall is due to the dramatic decrease in the possibility that mesh router i operates in checksum mode. As analyzed by our model, a mesh client speed faster than 0.05 x transmission range per second negates the possibility for a mesh router to operate in checksum mode and to decrease the control overhead by utilizing ADP. This result demonstrates the accuracy of the estimation made by our model on the control overhead. Simulation results in Figure 2.6 also confirm this. Moreover, the control overhead decrease shows that ADP saves bandwidth by optimizing control messages to distribute LAB.

◊ **Summary of Utilizing Association Discovery Protocol and MPR broadcasting**

In the architecture that we have proposed, the mesh routers are running not only a routing protocol to calculate their routing table but also an Association Discovery Protocol, whose task is to distribute the list of mesh clients associated with them. This Association Discovery Protocol had been integrated in an early draft of 802.11s specifications. We have described its principles, and some design decisions.

One of the challenges is to decrease the additional control overhead that is distributed using MPR-based broadcast protocol. This additional control overhead should not be ignored because a mesh router may associate with a large number of mesh clients, and in the proactive operation that was proposed, the mesh routers should know the entire list of mesh clients associated with other mesh routers. Thus the mesh router should (periodically) distribute information about all mesh clients associated with them. In this case, the additional overhead to distribute information may become larger than the overhead from the wireless mesh routing protocol. To answer this challenge, the proposed ADP divides association information into blocks, and can switch between two operating modes (per block). When part of its LAB is stable, a mesh router may issue only checksums to represent the content of its LAB, and refresh it for other routes (a checksum mode). Otherwise, it can still transmit full block information (a full mode). A version of the protocol was analyzed, where the switch between a full mode and a checksum mode is made for the entire set of blocks.

As the analytical model and simulation results demonstrate, this capability of transmitting checksums is reducing the additional control overhead of ADP while keeping LABs consistent in the entire WMN. More generally, the ADP proposed could be adapted to a wide variety of hybrid wireless networks.

2.3 Summary

In wireless networks, almost all protocols use a broadcast protocol, and hence the efficiency of the broadcast protocol is important. MPR-based broadcast protocol is one of the efficient broadcast protocols with store-and-forward routing. The MPR-based broadcast protocol improves the efficiency by limiting a set of forwarding nodes to an MPR set. The limitation results in decreasing the number of forwarding nodes. Because the forwarding number decrease results in an improvement in efficiency, the MPR-based broadcast protocol has been successfully used in many wireless protocols standardized in IETF (Internet Engineering Task Force) and IEEE (Institute of Electrical and Electronics Engineers). We extended the MPR-based broadcast protocol in wireless mesh networks for a draft of IEEE 802.11 standardization Group S. More precisely, we extended it to adapt an OLSR protocol for routing in wireless mesh networks, and proposed an association discovery protocol (ADP) in order to limit overhead of repeating information that is already broadcasted and locally stored in each node, such as a local database about mobile hosts. In addition, we analyzed the cost and the performance of executing them and presented a formula to adjust protocol parameters. With the parameter adjustment, control overhead to support communication in wireless mesh networks could decrease. The successful overhead decrease showed an efficient solution based on efficient broadcasting with store-and-forward routing specifically for control traffic that is delay-sensitive and partially changing over time.

However, the MPR-based broadcast protocol needs precise two hop neighbor information that could change because of link loss and node mobility. On the contrary, specifically for continuous traffic, there is one possible solution that does not require precise topology information but also theoretically outperforms other broadcast protocols with store and forward routing. It is network coding. Hence, we investigate the possibility of utilizing network coding for an efficient broadcast protocol in wireless networks.

Chapter 3

Network Coding Background

In this chapter, we first briefly introduce network coding in section 3.1. We then present a detailed review of recent research in the context of a practical framework to use network coding in section 3.2, and we explore the theoretical performance of network coding in section 3.3.

3.1 Introduction of Network Coding

The main advantage of network coding is that one transmission can bring multiple benefits by delivering useful information to several receivers at the same time. A key to enabling the multiple benefits is to permit coding at intermediate nodes. For any single source multicasting, Ahlswede et al proved that the multiple benefit is quantified as the maximum multicast capacity (a max flow and a capacity of min-cut) [8], which in some cases classical store and forward routing could not achieve. As a consequence, network coding can generally achieve higher throughput for multicasting than classical routing as demonstrated in the famous example, the butterfly network.

For instance, in the popular example of a butterfly network in Figure 3.1, we have a source s and two destinations $d1, d2$. We assume that each edge has unit capacity. As seen in Figure 3.1, both destinations $d1$ and $d2$ have two disjointed paths from the source s that are $s \rightarrow n1 \rightarrow d1$ and $s \rightarrow n2 \rightarrow n3 \rightarrow n4 \rightarrow d1$ for $d1$, and are $s \rightarrow n2 \rightarrow d2$ and $s \rightarrow n1 \rightarrow n3 \rightarrow n4 \rightarrow d2$ for $d2$. For each disjointed path, the source s sends two different bits $b1$ and $b2$. If one destination uses all available network capacity, the destination can receive the two bits $b1$ and $b2$ simultaneously. However, because two destinations

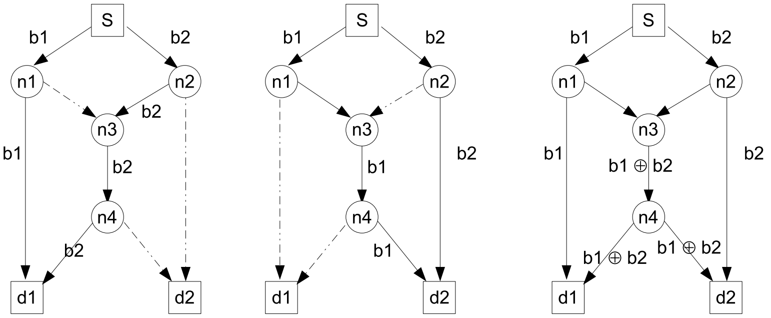


Figure 3.1: An example of a wired network with network coding

must share the network at the same time, classical routing cannot let both of destinations simultaneously receive two bits $b1$ and $b2$. The delay at one destination bounds the source rate for two destinations to one bit per unit time instead of two bits. The bound originates from the fact that two disjointed paths for each destination are overlapped at an edge between $n3$ and $n4$, and only one bit can pass by the overlapped edge. Namely, in store and forward routing, the overlapped edge becomes a bottleneck for the achievable source rate. However, the bottleneck can be solved by network coding that permits coding at the starting point of the bottleneck. When network coding is used, $n3$ in Figure 3.1 executes $\oplus$ for two received bits and transmits the result of $\oplus$ through the bottleneck edge. Then, $b1 \oplus b2$ arrives at each destination simultaneously with $b1$ or $b2$ respectively. The destination $d1$ extracts $b2$ from $b1 \oplus b2$ and obtains both bits, as does $d2$. In addition, when network coding is not used, available network resources are not fully utilized, as seen in this butterfly networks. When s sends $b1$ and $b2$ to $d1$ with classical store-and-forward routing, three dotted edges from $n1$ to $n3$, from $n2$ to $d2$, from $n4$ to $d2$ are not used. When s sends $b1$ and $b2$ to $d2$ with classical store-and-forward routing, three dotted edges from $n1$ to $d1$, from $n2$ to $n3$, from $n4$ to $d1$ are not used. But, with network coding all available network resources are fully used.

As seen in the example in Figure 3.1, network coding can improve throughput, decrease energy consumption with less transmission and also decrease the delay in receiving data. The advantages are essential to wireless networks, which are naturally unreliable, whose channels originally broadcasts data within a transmission range, and whose dominating equipment has energy limitations, such as mobile phones. Figure 3.2 shows a typical example where wireless networks naturally benefit from network coding.

In Figure 3.2, two nodes A and B communicate via a relay node R. To exchange a

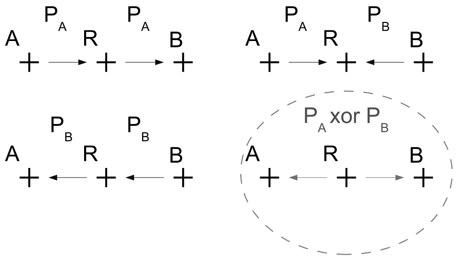


Figure 3.2: An example of a wireless network with network coding

packet with store-and-forward routing, one of the two nodes first sends a packet to the other through the relay node R, and the other node also sends its packet in a reverse direction. The exchange with store-and-forward routing needs four transmissions, but network coding can decrease the required transmission number into three. With network coding, each node sends its own packet to the relay node R first, and the relay node R generates $P_A \oplus P_B$ and broadcasts the result of $\oplus$. After $P_A \oplus P_B$ reaches at each node, the nodes can extract the intended packet. For example, node A extracts P_B and node B extracts P_A. Finally, the $\oplus$ operation decreases one transmission in exchanging data between node A and B.

3.2 Practical Framework of Network Coding

In this chapter, we explain the practical framework of network coding in order to use linear coding and random linear coding. A shorter instant primer on network coding can be found in [49].

3.2.1 Linear Coding

Network coding, as well as its benefits were first introduced by Ahlswede et al [8]. Ahlswede et al. [8] showed that with network coding, as symbol size approaches infinity, a source can multicast information at a rate approaching the smallest minimum cut between the source and any receiver. Since then, coding methods to achieve the multicast max-flow rates had been researched. Li et al. [50] showed that in single source multicasting, linear coding on a finite field (Galois field) is sufficient to achieve the individual max-flow rates between the source and any receiver. Koetter and Medard [51] presented an algebraic framework that extended previous results to unreliable networks, and proved the applicability of the min-cut max-flow bound for networks with delay and cycles.

Linear coding is a notable coding method, because it only uses a linear transformation, such as plus and multiplexing, which can be implemented with low computational cost. In order to apply the low complex linear coding on network packets, all packets are assumed to have an identical size, and the same sized packets are viewed as vectors over a fixed Galois Field (GF)$\mathbb{F}_q$. With this view of packets, a node linearly transforms incoming packets into outgoing packets like vectors.

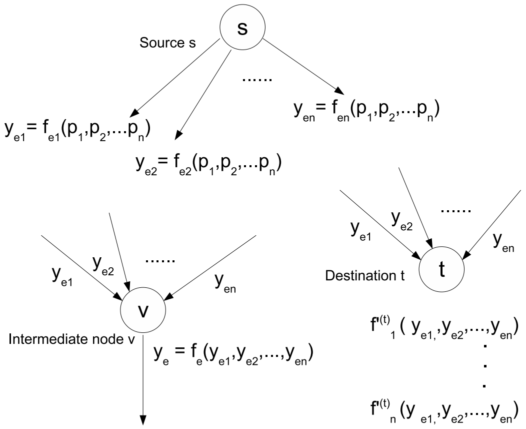

s
Source s
......
y_en= f_en(p_1,p_2,...p_n)
y_e1= f_e1(p_1,p_2,...p_n)
y_e2= f_e2(p_1,p_2,...p_n)
y_e1
y_e2
......
y_en
Destination t
t
y_e1
y_e2
......
y_en
v
Intermediate node v
y_e = f_e(y_e1,y_e2,...,y_en)
f^(t)_1 (y_e1,,y_e2,...,y_en)
f^(t)_n (y e1,,y_e2,...,y_en)

Figure 3.3: An encoding process

Figure 3.3 shows an example of network coding at the source s and an intermediate node v. More precisely, for a network coding setup, a network is represented as a directed graph $G = (V, E)$ where V is a set of nodes and E is a set of edges. We assume that each edge has a unit capacity and can transport a symbol in an alphabet of size q. A multicast session considers a source $s \in V$ and a set of destination nodes in $T \subseteq V$. A message consisting of the symbols $p_1, p_2, ..p_n$ is observed at the source s and is to be transmitted to all $t \in T$. Each intermediate node v produces output symbols (y_es) which are a function (f_e) of the input symbols $(y_{e1}, y_{e2}, ..., y_{en})$ at that node. Then, a network code is a specification of such functions f for all nodes as seen in Figure 3.3. The received symbols at destination t is also a function of the input symbols at the node. In this network coding process, the function used at each node is called a "local encoding vector" and the produced output symbol is called an "information vector". For example, f_e at intermediate node v is a local encoding vector of node v, and y_e is the result of using local encoding vector f_e at intermediate node v (a information vector).

In linear network coding, all local encoding vectors are linear functions over a finite field GF(q). For example, f_e at intermediate node v is : $f_e(y_{e1}, y_{e2}, y_{e3}, ..., y_{en}) = a_1y_{e1} + a_2y_{e2} + a_3y_{e3} + ... + a_ny_{en}$ where $a_1, ...a_n$ are chosen over GF(q). Then, by linearity, we can write the global transfer function from the source s to each destination node t as seen in Figure 3.4. Figure 3.4 also shows how the received packet at destination node t relates to the source packets. We call the coefficients $[g^t_{i1}, g^t_{i2}, ..., g^t_{in}]$ "a global encoding vector" for the encoded packet y^t_i for $1 \leq i \leq n$. Provided that $G^{(t)}$ in Figure 3.4 has rank n, the destination t can recover $P = [p_1, p_2, ..., p_n]$ by multiplying $Y^{(t)}$ by a left inverse of $G^{(t)}$: $G^{(-t)}Y^{(t)} = IP$.

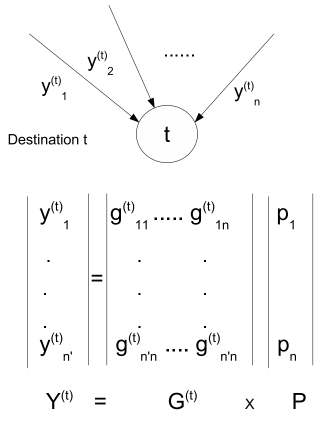


Figure 3.4: Received encoding data at destination t

The main benefit of linear coding is low complexity. The low complexity is emphasized as the size of GF becomes smaller. A smaller field makes the operations of linear coding much simpler. For example, when the field is $\mathbb{F}_2$, the packets are viewed as a sequence of bits and the coefficients are either 0 or 1. Thanks to the simple coefficients 0 or 1, the operations on vectors become an *exclusive or*, and both encoding and decoding becomes simple.

Although linear coding decreases the computation cost of all coding and decoding operations, we still have to decide a local encoding vector that must guarantee decoding at every node. Jaggi and Sanders et al presented polynomial algorithms to select the local encoding vector [52, 53]. However, the polynomial algorithms use global network topology and needs to inform each node which coefficient to use. Namely, the polynomial algorithms operate in a centralized way. In contrast to the deterministic codes using the centralized way, there exists a simple and fully decentralized method. The method is random linear coding.

3.2.2 Random Linear Coding

In order to use linear coding, we need to select coefficients, and polynomial algorithms [52,53] can be used to select these coefficients. The algorithms assign the valid network codes that guarantees decoding at every node, but they operates in a centralized and deterministic way. One notable coding method to operate in a fully decentralized way is *random linear coding*, which randomly selects its local encoding vector on a fixed Galois Field $\mathbb{F}_q$ without any coordination [9, 10]. For example, $a_1, a_2, ..., a_n$ are randomly chosen over GF(q) to comprise a local encoding vector of intermediate node v in Figure 3.4 for a encoded data $a_1 y_{e1} + a_2 y_{e2} + a_3 y_{e3} + ... + a_n y_{en}$.

With random linear coding, each node can randomly select a local vector for each outgoing packet, namely an information vector. This random selection causes a local encoding vector to continuously change even at the same node. Hence, it is hard to determine which local encoding vectors are sequentially used in order to generate the coded packet. If we do not know the local encoding vectors that are sequentially used to generate the coded packet, we cannot find a global encoding vector for the coded packet, and therefore decoding is impossible. Hence, we need a method to track the sequentially used local encoding vectors for eventual decoding. The method is packetization [9, 54].

The packetization in [9, 54] suggested inserting a global encoding vector in a special packet header and transmitting the global encoding vector with the encoded data. Let us assume that the source s sends n packets $(p_1, p_2, ..., p_n)$. When linear network coding is applied, each packet flowing in a network is a linear combination of the source packets as :

$$\text{i}^{th}\text{received coded packet at node } v : y_i^{(v)} = \sum_{j=1}^{j=n} g_{i,j}^v p_j$$

where the $(p_j)_{j=1,\ldots,n}$ are n packets generated from the source. The sequence of coefficients for source packets in a coded packet $y_i^{(v)}$ forms a global encoding vector $[g_{i,1}^v, g_{i,2}^v, \ldots, g_{i,n}^v]$.

As shown in this expression of a coded packet, a global vector contains the information how the received encoded packet $y_i^{(v)}$ is related with the source packets. Hence, a packet including n extra symbols for a global encoding vector ($[g_{i,1}, g_{i,2}, \ldots, g_{i,n}]$) in a special header with encoded data allows destinations to decode without knowing the network topology or the encoding rules in a completely decentralized way.

However, the packetization is not enough to practically use random linear coding. In [51] it is assumed that all incoming packets from all incoming edges arrive synchronously without delay. But in a real network, packets arrive asynchronously and depart with arbitrarily varying rates, delays and losses. To handle this real network condition, [9, 54] suggested using buffering as seen in Figure 3.5. A node saves incoming encoded packets and their global vectors delivered through its incoming edges. When a node should transmit a new coded packet, the node uses all or a portion of encoded packets saved in its local buffer and hence its incoming packets do not need to arrive synchronously at the node through its incoming edges.

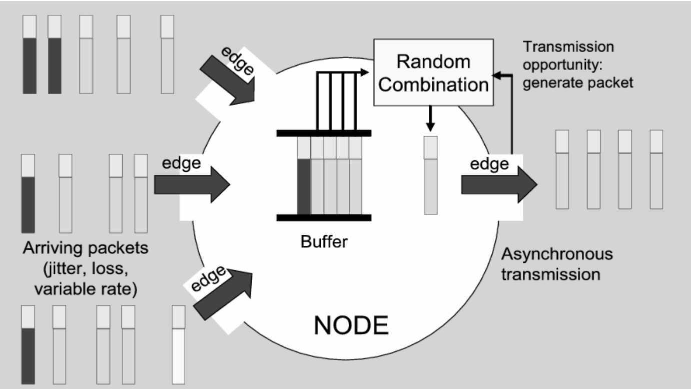


Figure 3.5: Buffering received packets (global vectors and information vectors)

The process to generate a new coded packet is almost same as the process explained in section 3.2.1; the difference is that it uses encoded packets (information vectors) and their global encoding vectors saved in its local buffer. More specifically, node v generates a newly encoded packet by linearly combining randomly chosen coefficients $[l_1, l_2, ..., l_{n'}]$ ($1 \times n'$ matrix) and the stored encoded packets $[[y^{(v)}_1][y^{(v)}_2]...[y^{(v)}_{n'}]]$ ($n' \times 1$ matrix) in v's local buffer where $1 \leq n' \leq n$. With the new coded packet, node v also generates a new global vector for the new coded packet by linearly combining the used coefficients $[l_1, l_2, ..., l_{n'}]$ ($1 \times n'$ matrix) and global vectors $[[g^{(v)}_{1,1}, g^{(v)}_{1,2} .., g^{(v)}_{1,n}], [g^{(v)}_{2,1}, g^{(v)}_{2,2} .., g^{(v)}_{2,n}], ..., [g^{(v)}_{n',1}, g_{n',2^{(v)}} .., g^{(v)}_{n',n}]]$ ($n' \times n$ matrix) saved in v's local buffer for saved encoding packets $[[y^{(v)}_1][y^{(v)}_2]...[y^{(v)}_{n'}]]$. These two vectors (new encoded data and new global encoding vector for the new encoded data) are delivered in the same packet by using packetization.

Thanks to the packetization and buffering, we can use random linear coding for practical networks, even for wireless networks which are unreliable and unstable.

In order to ultimately recover all of the source's packets, the rank of a node must be equal to the total number of source packets. This means that in the worst case, a node is supposed to store all received innovative packets in its local buffer until it can recover all source packets at once. In practice, the size of the local buffer in a node cannot infinitely increases, and hence we need to limit the number of encoding packets which is saved in a local buffer. The limitation of encoding packets is based on a *generation*. In other words, only packets related to the same set or the same generation of source packets $p_1, p_2, ..p_n$ can be encoded. In addition, the number of packets in a generation, n is called a *generation size* [54].

3.2.2.1 Decoding, Vector Space, and Rank

With packetization and buffering, a node stores received encoded packets in its local buffer and the stored encoded packets compose a matrix of a node v as seen in Figure 5.4. Let G_v denote global encoding vectors of the matrix of a node v. At any point of time, a node v is associated with the *vector space* spawned by the global encoding vectors G_v. The dimension of that vector space is denoted as D_v, $D_v \triangleq \dim \Pi_v$ and also as the *rank* of the matrix. In the rest of this thesis, that rank and dimension is called the *rank of a node*. The rank of a node is a direct metric for the amount of useful received packets. When a received packet increases the rank of the receiving node, the received packet brings useful information and is called *innovative*. On the contrary, if a received packet does not increase the rank of the receiving node, the packet is not useful and is non-innovative. The non-innovative packet contains redundant information and cannot contribute to recovering the source packets. Thus, depending on whether a received packet is innovative or not, the receiving node can decide to drop or store the received packet. If the matrix of the node $G^{(v)}$ has full rank(n) or a submatrix with full rank($r < n$) exists, the node can decode n or r source packets at

$$\underset{Y^{(v)}}{\begin{vmatrix} y^{(v)}_{1} \\ y^{(v)}_{2} \\ \ldots\ldots \\ y^{(v)}_{n} \end{vmatrix}} \underset{=}{=} \underset{G^{(v)}}{\begin{vmatrix} g^{(v)}_{1,1} & g^{(v)}_{1,2} & \cdots & g^{(v)}_{1,n} \\ g^{(v)}_{2,1} & g^{(v)}_{2,2} & \cdots & g^{(v)}_{2,n} \\ & \cdots\cdots\cdots & & \\ g^{(v)}_{n,1} & g^{(v)}_{n,2} & \cdots & g^{(v)}_{n,n} \end{vmatrix}} \underset{\times}{} \underset{P}{\begin{vmatrix} p_1 \\ p_2 \\ \cdots \\ p_n \end{vmatrix}} \qquad \underset{G^{-(v)} \times Y^{(v)}}{G^{-(v)} \begin{vmatrix} y^{(v)}_{1} \\ y^{(v)}_{2} \\ \ldots\ldots \\ y^{(v)}_{n} \end{vmatrix}} \underset{=}{=} \underset{I}{\begin{vmatrix} 1 & 0 & 0 & .. & 0 \\ 0 & 1 & 0 & .. & 0 \\ & & \cdots\cdots\cdots & & \\ 0 & 0 & 0 & .. & 1 \end{vmatrix}} \underset{\times}{} \underset{P}{\begin{vmatrix} p_1 \\ p_2 \\ \\ p_n \end{vmatrix}}$$

Figure 3.6: A decoding process at a node v

the same time. The decoding amounts to inverting the matrix or submatrix, for instance, using Gaussian elimination

3.3 Theoretical Achievable Performance of Wireless Network Coding

In this section, we describe the theoretical performance of broadcasting with network coding in terms of a maximum achievable broadcast rate. The theoretical results about the performance of network coding have been focused on multicasting. Because broadcasting is just a special case of multicasting, the theoretical results for multicasting are also valid for broadcasting. Therefore, from here we review a body of literature on network coding performance for multicasting, and then directly use the theoretical results of multicasting for broadcasting.

The bound of the maximum multicast capacity with network coding is theoretically the multicast min-cut. The theoretical bound implies that network coding enables multicasting to have the max-flow min-cut bound. In fact, the max-flow min-cut bound for unicasting was already proved in 1956 by Elias et al [55] and Ford and Fulkerson [56], and store-and-forward routing without network coding also can achieve the bound for unicasting. However, in the case of multicasting, there are some cases that store-and-forward routing cannot achieve the max-flow min-cut bound. In these cases, the max-flow min-cut bound for multicasting can be achieved by network coding [8] and also for wireless multicasting if the min-cut is computed over a hypergraph.

In order to explain the theoretical maximum capacity (rate) for wireless broadcasting with network coding, this section first explains the maximum unicast capacity. Then, we apply the same concept to multicasting, and next explain why the achieved maximum multicast capacity with network coding could be higher than the achieved maximum multicast capacity with store-and-forward routing. Finally, we describe a maximum broadcast rate for wireless network coding and the effects of using random linear coding.

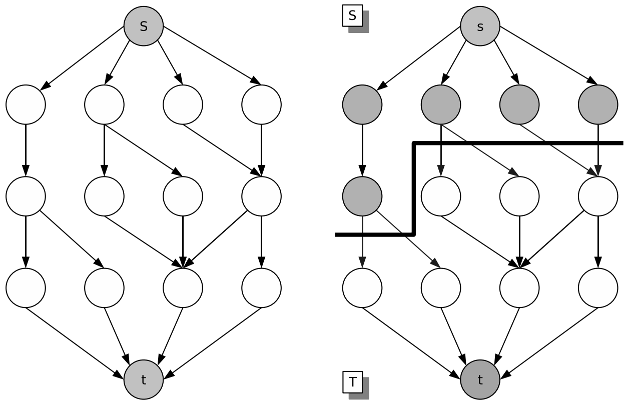
3.3.1 Min-cut Max flow Theorem

For unicasting between a source and a destination, the maximum flow is bounded by the min-cut with the min-cut max-flow theorem [55–57]. This means that the source cannot send faster than the bound, the min-cut, and the min-cut becomes a maximum unicast rate that the source can transmit. Before explaining why the upper bound of a unicasting rate is the min-cut, we first describe the basic concepts of network flow in graph theory.

In graph theory, a network is a directed weighted graph whose edges have a non-negative capacity. Formally, a network is defined as: A finite directed graph $G = (V, E)$ where V is a set of nodes(vertex) and E is a set of edges. In G, every edge $(u, v) \in E$ has a non-negative, real-valued capacity $C(u, v)$. If edge $(u, v) \notin E$, the capacity of the edge (u, v) is assumed to be 0. We distinguish two vertices: a source s and a sink t. For a source s and a sink t, we assume that there is a path $s \rightarrow v \rightarrow t, \forall v \in V$. A flow in G is a real-valued function $f : V \times V \rightarrow \mathbb{R}$ with the following three properties for all u and v in V.

- capacity constraints : $\forall u, v \in V, f(u, v) < C(u, v)$. The flow along each edge cannot exceed its capacity.
- skew symmetry: $\forall u, v \in V, f(u, v) = -f(v, u)$. The net flow from u to v must be the opposite of the net flow from v to u.
- flow conservation: $\forall u \in V - \{s, t\}$, $\sum_{u \in V} f(u, w) = 0$. The network flow to a node is zero except for the source s producing flow and the sink t consuming flow.

$f(u, v)$ is called the flow from u to v, and can be positive, negative or zero. The value of a flow f is defined as total flow out of the source: $|f| = \sum_{v \in V} f(s, v)$. When a flow has the largest value, we call the flow *a maximum flow*. The maximum flow provides the largest possible total flow from the source s to the receiver t in a given directed graph G. Hence, the source cannot transmit data faster than the maximum flow and the maximum rate of the source is equal to the maximum flow.



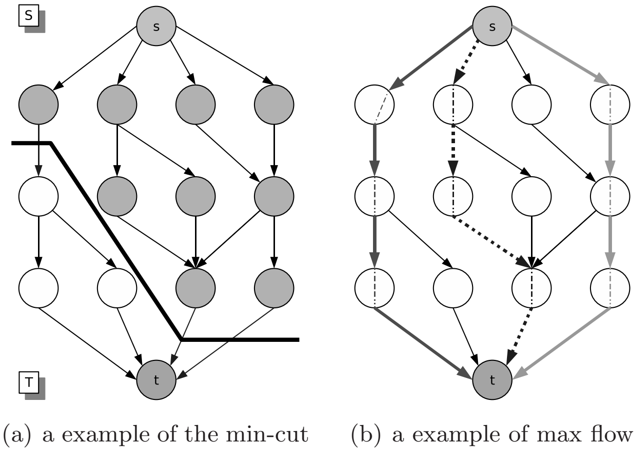
(a) an example of network (b) An example of a cut

Figure 3.7: An example of S/T partition and a cut

Now, we briefly explain why the maximum flow between s and t is bounded by a minimum cut between s and t. To explain how the minimum cut bounds the maximum flow, let us look at the definition of a cut. A cut of a graph $G = (V, E)$ is a partition of the vertices V into two sets S and T. Any edge $(u, v) \in E$ with $u \in S$ and $v \in T$ is a cut edge and the sum of weights of the cut edges decides the capacity of the cut, as seen in Figure 3.7. Hence, the capacity of an s-t cut is the sum of the capacity of all edges crossing the cut from the region S to the region T. Formally:

$$C(S,T) = \sum_{(s,t)\in E, s\in S, t\in T} C(s,t) \tag{3.1}$$

In Figure 3.7, Figure 3.7(b) represents one example of a cut of graph in Figure 3.7(a).



(a) a example of the min-cut (b) a example of max flow

Figure 3.8: An example of the min-cut and the max flow

With capacity constraints, for any s-t cut and flow, the capacity of s-t cut is the upper-bound of the flow across the s-t cut, $C(S,T) \geq f(S,T)$. With skew symmetry and flow conservation, for any s-t cut, the value of flow across the cut equals the value of the flow, $|f| = f(S,T)$. Therefore, for any s-t cut and any flow, the capacity of s-t cut is the upper-bound of value of the flow, $|f| \leq C(S,T)$. Because the upper bound is a bound for any value of flow including a maximum flow, the minimum value of all possible s-t cuts becomes the upper bound of a maximum flow as seen in Figure 3.8. Namely, the min-cut $C_{\min}(s,t)$ with the minimum capacity among all s-t cuts bounds a maximum flow. When $Q(s,t)$ denotes the set of all s-t cuts , the min-cut $C_{\min}(s,t)$ is defined formally as:

$$C_{\min}(s,t) \triangleq \min_{(S,T)\in Q(s,t)} C(S) \tag{3.2}$$

Figure 3.8(a) shows the min-cut of the example network in Figure 3.7. The min-cut is the same with the maximum flow through three disjointed paths in Figure 3.8(b). We explain it with the Menger Theorem again.

The max-flow min-cut theorem formally states the relationship between the min-cut and the max-flow as:

- Max-Flow Min-Cut theorem : Let $G = (V, E)$ be a directed graph, and let u and v be vertices in G. The maximum weight among all (u, v)-flows in G equals the minimum capacity among all sets of edges in E whose deletion destroys all directed paths from u to v.

The upper bound of unicasting, $C_{\min}(s, t)$ can be achieved also by store-and-forward routing without network coding. In order to find a maximum unicast rate, store-and-forward routing uses low-order polynomial-time algorithms which are augmenting path algorithms such Ford-Fulkerson [56]and its variants [58, 59], or Push/relabel algorithms [60]. Unlike unicasting, store-and-forward routing could not achieve the upper bound, i.e., the multicast min-cut in some cases. For the cases, only network coding can achieve the upper bound for multicasting. In the next section, we explain why network coding could achieve the bound for multicast in these cases and store-and-forward routing could not.

3.3.2 Maximum Multicasting Rate of Network Coding

The maximum capacity of a flow is bounded by the min-cut, and hence the min-cut becomes the upper bound of a maximum rate for the source to transmit data. For unicasting, classical routing also can achieve the upper bound, the min-cut with the min-cut max-flow theorem. However, on some cases the upper bound for multicasting could not be achieved without network coding. To explain why classical routing could not achieve the upper bound of multicasting, we first compute the multicast min-cut. Because multicasting has several receivers, computing the multicast min-cut needs to consider all individual min-cuts for each receiver and to take the minimum value among all individual min-cuts. Formally:

$$C_{\min}^{\text{all}}(s) \triangleq \min_{t \in \mathcal{V} \setminus \{s\}} C_{\min}(s, t) \tag{3.3}$$

The relationship between the min-cut and the max-flow can also be explained with the Menger Theorem [57] stating that a maximum rate and the min-cut $C_{\min}^{\text{all}}(s)$ satisfy the constraints on edge-independent paths. Formally;

- Edge-connectivity version of Menger Theorem : Let $G = (V, E)$ be a directed graph, and let u and v be two nonadjacent vertices in G: $u, v \in V$. Then, the size of the minimum edge cut for u and v (the minimum number of edges whose removal disconnects u and v) is equal to the maximum number of pairwise edge-independent paths from u to v.

For simplicity, let us assume that a directed graph $G = (V, E)$ with unit capacity edges as [61]. If the minimum cut between s and t equals h on the graph, the maximum amount of information that can be sent from s to t per unit time equals to h with the max-flow min-cut theorem. With the Menger Theorem, the min-cut and the max-flow also means that there exist exactly h edge-disjointed paths between s and t. The example in Figure 3.8 shows the relationship between the min-cut and the number of disjointed paths for unit capacity edges by showing that the min-cut is three and that there exist three disjointed paths between s and t (a thick line, a dotted line, and a gray line)

For multicasting, there exist several receivers, and hence with the Menger Theorem for each receiver, there exist h edge-disjoint paths for each receiver. However, although h paths for a receiver are edge-disjoint for each pair, Menger Theorem does not assure that all paths of h x the number of receivers are edge-disjoint. In other words, when multiple receivers use a network simultaneously, the h disjointed paths for each destination may be overlapped. Then, receivers must share the capacity of theses overlapped pathes and sharing capacity decreases the capacity for each receiver. The decreased capacity for a receiver results in reducing the min-cut of the receiver. The reduced min-cut of a receiver causes reduction of the multicast min-cut, and finally results in a reduction of the maximum multicast capacity and a maximum multicast transmission rate for the source.

Unlike store-and-forward routing, network coding can prevent the capacity division caused by overlapped paths. With network coding, intermediate nodes transmit mixed data from different incoming flows, and then each receiver that shares the same edge can obtain new information for itself as if only that receiver accessed the network. As a result, receivers do not need to share capacity of the overlapped paths, and a maximum multicast rate is exactly equal to the multicast min-cut, that is the minimum value of min-cuts to every receiver. Finally, we have the max-flow min-cut theorem for multicasting of [8] as:

- A multicast generalization of the max-flow min-cut theorem: The maximum rate at which information can be simultaneously multicasted from a source to all sinks is equal to the minimum capacity of a cut from a source to all sinks.

3.3.3 Maximum Broadcast Rate of Wireless Network Coding

In section 3.3.1 we explained that the multicast min-cut bounds a maximum multicast rate and only network coding can achieve the max-flow min-cut bound for multicasting. Because broadcasting is a kind of multicasting except that all nodes are destinations, the maximum broadcast rate is also the minimum value of min-cuts to all nodes other than the source.

Although our target networks are wireless networks, we explain the min-cut and a maximum broadcast rate in wired networks that are modeled as a directed graph. In this section, we compute them for wireless networks. In order to compute them in wireless networks, we should consider a directed hypergraph that models a wireless network by reflecting the feature of wireless media where transmitting data reaches all nodes within a transmission range. To define the min-cut over a hypergraph, we use the same notation over a graph; $Q(s,t)$ denotes the set of all s-t cuts on a hypergraph. As defined in a general graph, the capacity of an s-t cut is the sum of the capacity of all edges crossing the cut from the region S to the region T. The only difference in wireless networks is that the crossing edges are hyperedges where one transmission from a node in S can reach several nodes in T. Considering the feature of hyperedges, we define the capacity of the cut in terms of neighborhood; the capacity of the cut is defined by a set of nodes in S that have at least one node in T for a neighbor as seen in Figure 3.9. We denote the set of nodes in S as ΔS ($u1$ and $u2$ in Figure 3.9). Then the *capacity of the cut* $C(S)$ is defined as the sum of rates from the nodes in ΔS.

$$\Delta S \triangleq \{v \in S : \mathcal{N}(v) \cap T \neq \emptyset\} \quad \text{and} \quad C(S) \triangleq \sum_{v \in \Delta S} C_v \tag{3.4}$$

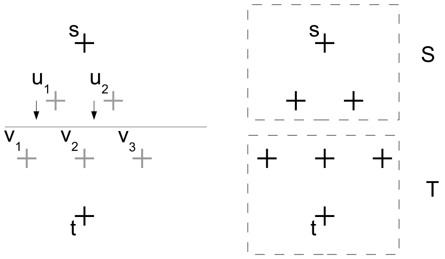


Figure 3.9: an example of a cut capacity

With the same logic used in a general graph, a max flow for multicasting in a hypergraph is bounded by the minimum of min-cuts for all destinations. Therefore, a maximum broadcast rate $C_{\min}^{\text{all}}(s)$ with network coding for the source s in wireless networks is the minimum of min-cuts as $C_{\min}^{\text{all}}(s) \triangleq \min_{t \in \mathcal{V} \setminus \{s\}} C_{\min}(s,t)$.

3.3.4 Maximum Broadcast Rate of Random Linear Coding

Since Ahlswede et al proved that network coding can achieve a maximum multicast rate that is the multicast min-cut, efficient coding methods to achieve the maximum multicast rate have been researched, and linear coding was proved to be sufficient to achieve the maximum multicast rate for any single source multicasting [50]. Linear coding only uses linear transformation over fixed Galois Field (GF) $\mathbb{F}_q$, and hence has low complexity. An algebraic framework for linear coding is also presented in order to study the network capacity [51]. In addition, it is proved that there exists a linear network coding solution for a single source multicasting over a finite field $\mathbb{F}_2^m$ for some large enough m whose bound is $m \leq \lceil \log_2(RC_s + 1) \rceil$ where R is the number of receivers and C_s is a source rate [51].

The proof of low complex linear coding is followed by the novel work of [11, 62] presenting a fully decentralized coding method, random linear coding. We could regard random linear coding as a kind of generalization of fountain code. However, random linear coding for network coding is different from fountain code, in which fountain code may encode only at the source node, while network code may execute encoding at every node in a network. An important feature of random linear coding is that it can achieve the optimal throughput, the max-flow min-cut bound for multicasting. This important feature was proven in synchronized networks [63] and also in non synchronized networks regardless of scheduling [9]. For the proof in synchronized networks, Jafarisiavoshani et al [63] assumed that at time t a node would send a linear combination of packets received until time $t-1$ with random coefficients. Then, after a transitional phase, every node could receive innovative packets at a rate (exactly, the multicast min-cut). For the proof in non synchronized networks, Ho et al [9] assumed that a node would collect received encoded packets for some delay, and then the node would send a linear combination of collected packets with random coefficients. Then, distribution of delay at each node could define an average rate of each node. After the multicast session would be executed long enough for the average rate to become stable, a maximum broadcast rate of the source could become the lowest capacity of the min-cut to any destination considering the average rate. In other words, a source could achieve the maximum multicast capacity (the multicast min-cut) with arbitrary scheduling. These results could be asymptotically true also in lossy networks because the average rate implicitly includes loss.

In addition, the work of [11] stated that at the end of a multicast session, all the destinations could decode with a probability p_e. The error probability p_e may become arbitrarily small as the field size increases : $p_e = 1 - \frac{1}{|F|}$, where $|F|$ is the cardinality of the finite field of coefficients. Moreover, the recent work of [19] stated that the error probability p_e may also be independent of the field size when a multicast session executes sufficiently long time. More specifically, the small sized field increases the number of non-innovative packets and the increased non-innovative packet number increases a gap of ranks between neighbors. The increased gap increases the probability that an incoming encoded packet is an innovative packet, and finally the new encoded packet from neighbors with a higher rank than the receiver is asymptotically innovative as if the packet is encoded over large sized GF. Namely, the large gap of ranks between neighbors works like the large sized GF. If a multicast session is executed for a long time, the starting phase, when a gap of ranks between neighbors increases, and the ending phase, when the gap decreases, are relatively negligible comparing the whole session duration. Therefore, random linear coding asymptotically guarantees not only a valid coding but also the optimal throughput achievement, regardless of a fixed GF size.

3.4 Summary and Problem Statement for Efficiency Broadcasting with Network Coding

In this chapter, we explained a practical framework to use network coding and theoretically achievable performance of network coding. The theoretical upperbound of achievable broadcasting capacity is the max-flow min-cut bound that is optimal capacity, and that store-and-forward routing could not achieve in some cases. In addition to the capacity improvement, network coding can bring other benefits like energy efficiency and short delay as shown in the two examples in section 3.1.

These advantages are necessary to wireless networks and hence it is natural to use network coding for efficient wireless broadcasting that is indispensable for all wireless protocols where the efficiency is measured by the transmission number of all nodes in order to send data from a source to the whole network. Finding an optimal solution for efficient broadcasting in wired networks is usually finding a minimum spanning tree from the source to all destinations, and many algorithms such as Prim's algorithm, Kruskal's algorithm, and the Reverse-Delete algorithm have been proposed. With these algorithms, store-and-forward routing can execute multicasting or broadcasting with the minimized cost in wired networks. On the contrary, it is NP-complete to find the optimal broadcasting (minimum energy broadcasting) in wireless networks [64] where each transmission uses all links of the emitter and the cost is attached to nodes instead of links. Because of the difficulty to perform wireless optimal broadcasting, many heuristics have been researched for efficient wireless broadcasting.

However, with network coding, we can find an optimal solution for wireless broadcasting in polynomial time. In fact, network coding can solve the extended case of broadcasting, multicasting in a polynomial time. The polynomial optimal solution with network coding is composed of two parts: coding and cost optimization. A coding part takes responsibility for assigning valid network codes to nodes that guarantee that all nodes can definitely recover all source packets. A cost optimization part takes responsibility for selecting nodes that inject coded packets and their transmission rates.

For the coding part, there exists an efficient and fully decentralized coding method as explained in section 3.2. It is random linear coding [9, 10], which regards a packet as a vector and randomly chooses coefficients for the vector on a fixed Galois Field. In addition, the novel work of [11] proved that random linear coding achieves a maximum multicasting rate on the condition that multicasting has only one source and the multicast session is executed sufficiently long. In addition, random linear coding enables every destination to decode with high probability regardless of a GF size. Thanks to this result, just by using random linear coding, we can be sure to achieve a maximum multicast rate with low error probability.

When the novel result of random linear coding is utilized, the remaining problem for efficient broadcasting is cost optimization. Because our aim is broadcasting where all nodes execute coding, the cost optimization for broadcasting based on random linear coding can be translated into selecting transmission rates of all nodes, namely *rate selection*.

Hence, a key of efficient broadcasting with random linear coding is formally defined as:

- The problem statement : How to select a rate of each node for efficient broadcasting with network coding.

The rate selection problem can be formulated in a linear optimization problem with polynomial variables that can be solved in a polynomial time [12, 13, 65] and possibly in a distributed way [13]. Although an optimal rate selection solution can be found in polynomial time by solving a linear optimization problem, the linear optimization problem has approximately N^2M variables where N is the total number of nodes and M is the average number of neighbors. This means that the worst case complexity bound to solve it using simplex method wit most efficient Karmarkar algorithm [66] is $O(N^7M^{3.5})$. In addition, it takes more time and more memory to solve the linear optimization problem as the network becomes larger and denser. Hence, we naturally ask if simple rate selection can achieve the optimal efficiency, or can achieve sufficiently good efficiency even if the efficiency is not optimal. As an answer to the question, we present our approach that theoretically achieves asymptotical optimality in large dense networks, and experimentally outperforms other broadcasting with store-and-forward routing.

Chapter 4

Efficient Wireless Broadcasting with Network Coding

4.1 Overview

In this chapter, we focus on using network coding as an efficient method to broadcast data to the whole network, where the cost is the total transmission number. Finding an optimal solution with classical store-and-forward routing for energy efficient broadcasting in wireless networks is known to be NP-complete. However, with network coding we can find the optimal solution in polynomial time. The polynomial solution with network coding has two parts: a cost minimization part that decides a rate of each node to minimize the sum of the rates and a coding part that finds valid network codes to guarantee decoding at all nodes.

For the coding part, results from information theory suggest a notable coding method, random linear coding, which can execute encoding and decoding in a fully distributed way without any coordination, and still guarantees with high probability that every node can recover the original packets from the source. Along with the simplicity, random linear coding also guarantees with high probability to achieve the bound of a maximum multicast rate if the multicast session is executed sufficiently long [11,62]. To take advantage of these novel results, our broadcast protocol uses random linear coding.

When random linear coding is used, a remaining problem is to minimize the cost for efficient broadcasting, and to minimize the cost is essentially deduced to find the average transmission rates of the nodes - the average rates are established on the entire duration of the stream (even for packet networks, see [11] for a recent synthesis of such results). As as result, finding an optimal solution to minimize the cost requires finding the optimal average rate of every node [11, 62]: " rate selection". The problem is then reformulated:

- problem statement : how to find a simple rate selection algorithm to achieve good performance?

In fact, [12, 13, 67] have shown that an optimal rate selection solution can be obtained by solving a linear optimization problem in a polynomial time and possibly in a distributed way [13]. However, a different and even simpler approach is possible. We introduce the approach in this chapter.

Our approach started from the investigation of maximal-efficiency-benefit of wireless network coding. Considering that a coded packet can bring new information to multiple receivers at the same time, the maximal-efficiency-benefit of wireless network coding can be achieved to make every transmission bring new information to all receivers. For a practical solution, our approach intends to achieve the maximal-efficiency-benefit of wireless network coding with a simple rate selection method.

This chapter is organized in the following way: we first review literature to decrease the cost of protocols with network coding in section 4.2, explain a network model and efficiency evaluation metrics in section 4.3.2,introduce a principal rationale with basic operations to use random linear coding in section 4.4. After that, we theoretically analyze the efficiency in section 4.5, we present experimental results in section 4.6, we propose improvement for spare networks in section 4.7 and we summarize in section 4.8.

4.2 Related Work of Efficient Wireless Network Coding

In this section, we review literature related with cost minimization of protocols based on network coding where the cost is measured by the total transmission number to deliver one data unit.

MORE: Efficient Multicasting with Opportunistic Routing

For multiple unicasting, a node could opportunistically decide whether the node transmits the new encoded packet or not, and which packets should be encoded [68, 69]. The opportunistic decision refers to the overheard information about received data by the neighbors. Chachulski et al extended the opportunistic routing protocol for multiple unicasting to a single source multicasting routing protocol, MORE [17] by combining the opportunistic routing with ExOR [70] and ETX [71]. In a MORE protocol, a header of a new encoded packet has a forwarder list including nodes closer to destinations than the transmitter of the new encoded packet. When a node receives the innovative coded packet, the node checks a forwarder list stored in the received coded packet. If the node finds itself in the forwarder list, the node is indicated to generate a new coded packet. To decide a new coded packet generation, the node checks its own credit value that indicates the encoded packet that the node can generate. A positive credit permits the node generating a new encoded packet and transmitting it, but a negative credit forbids it. After a node generates a new coded packet, the credit of transmitter decreases by one. When a node receives an innovative packet, the credit of a node increases by TX_{credit} . The TX_{credit} of a node is based on two factors: the first factor is the coded packets number transmitted by nodes that are farther from the destination than the node. The second factor is the loss probability between the node and the nodes farther from destination than the node. Precisely:

- $TX_{crediti} = \frac{z_i}{\sum_{j>i} z_j(1-eji)}$,
  ◇ z_j is the number of transmissions made by nodes (node j) which are farther from the destination than node i
  ◇ eji is loss probability from the farther nodes (node j) to node i.

With the above rule, a node that is located closer to the destination than a transmitter generates as many new encoded packets as intermediate nodes generates between the transmitter and the source. As a result, a node heuristically decides the transmission number depending on the relative distance to destinations and loss probability. The decision by the heuristics improves the throughput of multicast 35 - 200 percent, but the heuristic needs precise global topology and loss probability between each node. Because the global information may change, each node should continuously update the global information. When the global information is incorrect, the efficiency of the protocol could decrease. However, our approach does not need to know global information, or even precise local information, such as the exact two hop neighbor list. Our approach just uses the size of the neighborhood of each neighbor and the average number of neighbors inside a network. Even though the average number of neighbors is a kind of global information, the average number of neighbors usually remains steady and changes with small variation. Hence, we can say that the control overhead for our approach is relatively negligible.

Energy Efficient Broadcasting for all-to-all Broadcasting

For broadcasting from all nodes to all nodes, Fragouli et al suggested the deterministic algorithms that decided whether a node transmitted new encoded packets or not upon receiving an innovative packet [72, 73]. In [72], the decision is made using a sending count s that is initialed as zero, increases by the forwarding factor d on new innovative reception, and decreases by d on sending a new encoded packet. When a node has positive s, the node is allowed to transmit a new encoded packet. The subsequent paper [73] suggests heuristically adjusting the forwarding factor d with a two hop neighbor number as :

- $d_v = \frac{k}{min_{v' \in N(v)} |N(v')|}$, where $N(v)$ denotes the set of direct neighbors of node v and k is a forwarding factor.

In the above formula for forwarding factor d, variable k is used. Through modelling, Fragouli et al suggested forwarding factor k of more than three would guarantee decoding at every node for this all-to-all broadcasting. Their experimental results and specific examples show that lower values of k are possible. Inspired by their work, our approach also uses the size of the neighborhood of each neighbor. However, unlike their works, all-to-all broadcasting in [72, 73], our target is one-to-all broadcasting, and hence we could utilize the results of information theory, which shows that multicast max-flow rate for one source to all destinations is a multicast min-cut. We explain our different approach, which is inspired by [72, 73] in section 4.4.

Optimal Multicasting with Linear Optimization

The above two protocols (MORE [68, 69] and all-to-all broadcasting [72, 73]) let a node decide the transmission number proportional to the number of received innovative packets. For single source multicasting, we can use the novel result of [11], which states that single source multicasting with random linear coding can achieve a maximum multicast rate if a multicast session is executed sufficiently long. The novel result means that each node can achieve a maximum multicast rate just by generating a new encoded packet with random linear coding whenever possible. However, the blind execution of random linear coding can consume all available network capacity in order to achieve the maximum multicast rate. Hence, we need a solution to minimize the cost while achieving the maximum multicast rate.

The cost minimization (optimization) problem is a linear optimization problem to achieve a maximum multicast rate with the lowest cost. That is, for a given $G = (V, E)$, a source s

and destinations $t \in T$, the cost minimization problem is formulated as a linear optimization problem as:

- minimize $f(z) = \sum_{i|i \in V} \sum_{j|(i,j) \in E} z_{ij}$
  subject $z \in Z$,
  $z_{ij} \geq x_{ij}^{(t)} \geq 0, \forall (i,j) \in E, t \in T.$
  $\sum_{j|(i,j) \in A} x_{ij}^{(t)} - \sum_{j|(j,i) \in E} x_{ji}^{(t)} = \sigma_i^{(t)}$,
  where
  $$\sigma_i(t) = \begin{cases} R & \text{if } i = s, \\ -R & \text{if } i = t, \\ 0 & \text{otherwise.} \end{cases}$$

The vector z is part of a feasible solution for a linear optimization problem if and only if there exists a network code that sets up a multicast connection in the graph $G = (V, E)$ where V is the set of a nodes and E is the set of edges. In the multicast session, an average rate from the source s to all destinations in the set T is arbitrarily close to R and a flow on each link (i, j) is arbitrarily close to z_{ij}. For wireless networks, we consider a hypergraph and a hyperedge instead of a normal graph and an edge.

Because the above linear optimization problem has the polynomial number of constraints, we can solve it in a polynomial time [12, 13, 65], and even in a distributed way [12, 13]. By contrast, the same problem using classical store-and-forward routing is NP-complete.

However, although we can obtain an optimal solution in a polynomial time, the average (smoothed) complexity to solve a linear optimization problem is $O(N^3M)$ if we solve the linear programming with a simplex method [74]. The complexity $O(N^3M)$ is deduced from the expected complexity of a simplex method. The expected complexity is $O(p^2q)$ where p is the number of constraints and q is the number of variables. The optimal problem for wireless broadcasting with network coding has N^2M variables where N is the total number of nodes and M is the average number of neighbors, and N constrains. Hence, the complexity to obtain the optimal solution is $O(N^3M)$. In addition, the best-known-bound of the complexity to solve the problem is $O(q^{3.5})$ with Karmarkar's algorithm [66], that is $O(N^7M^{3.5})$. The complexity may not be low. In addition, it takes more time and more memory to solve the problem as a network becomes larger and denser.

Moreover, the decentralized method obtains an optimal solution by using Lagrangian dual of a linear optimization problem and a subgradient method, and hence it needs sufficient interaction in order to converge the solution to be optimal.

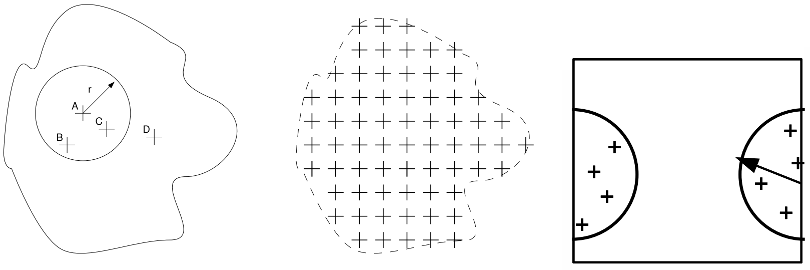
Instead of solving a linear optimization problem, we propose a different and simpler approach that is efficient and practical. Our approach is asymptotically optimal in large dense networks and could outperform other broadcasting with classical store-and-forwarding in practical networks.

4.3 Network Model and Evaluation Metrics

4.3.1 Network Model

A network that we assume is an ideal wireless model, where wireless transmissions are done without loss, collisions or interferences, and where each node of the network is operating well below its maximum transmission capacity.

Such networks specifically on 2 dimensional Euclidean spaces have been modeled as *unit disk graphs* [75], where two nodes are neighbors whenever their distance is lower than a fixed transmission range; see Figure 4.1(c) for the principle of unit disk graphs.



(a) Unit disk graph - neighbors of A are B and C since they are within range ρ

(b) Lattice

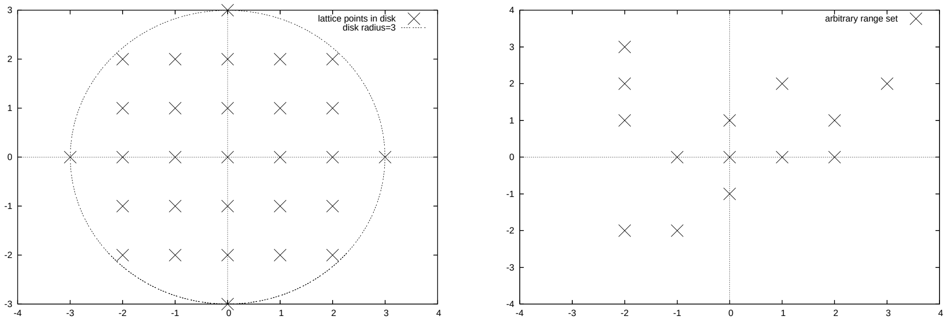
(c) Unit disk graph on torus

Figure 4.1: Network Models

- Random unit disk graphs with nodes uniformly distributed (Figure 4.1(a))
- Unit disk graphs with nodes organized on a lattice (Figure 4.1(b))
- Lattice networks where the neighborhood of one node is not necessarily the set of nodes within a disk, like in Figure 4.2(a), but may be any arbitrary set R such as the

one in Figure 4.2(b). it is a special case of unit disk graphs.

For lattice networks, the set R is fixed for one origin node, and all nodes of the lattice networks have a similar neighborhood by translation. In addition, we also consider variants of the random unit disk graph and the lattice graph where the network is a torus, with wrap-around connections in both the x and y directions as in Figure 4.1(c).



(a) Disk range for a lattice graph

(b) Arbitrary "range" for a lattice graph

Figure 4.2: Examples of range for lattices

In addition, we use hypergraphs to model wireless networks because hypergraphs can model the wireless transmission where the same transmission can reach several neighbors simultaneously as formalized in [76, 77] and [78].

4.3.2 Evaluation Metrics and Efficiency Bound of Store-Forward Broadcasting

The goal of this thesis is to design efficient broadcasting with network coding in wireless networks. Here, efficiency refers to energy-efficiency, which is measured by the transmission number to deliver one data unit, for example a packet, to the whole network. In order the evaluate the energy efficiency of our protocol, we compute E_{cost} of our protocol as the transmission number of a protocol to broadcast one unit data to the whole network. Then, For comparison, we provide two kinds of efficiency references: references with network coding and references without network coding. The efficiency references with network coding are $E_{optimal}$ and E_{bound}. $E_{optimal}$ is the optimal transmission number for broadcasting with network coding, and is obtained by solving a linear optimization problem, and E_{bound} is the minimal transmission number when maximal-efficiency-benefit of wireless network coding is achieved for broadcasting. The efficiency references with network coding are $E^{(\text{no}-\text{coding})}_{\text{rel}-\text{cost}}$,

E_{cost} of MPR-based broadcasting, and E_{cost} of CDS-based broadcasting that form CDS using greedy algorithm with global topology. Details of these references are described in section 4.3.2.1 and section 4.3.2.2.

4.3.2.1 Evaluation Metrics of Efficiency with Network Coding

The transmission number to broadcast one data unit to the whole network is denoted by E_{cost}. E_{cost} of network coding broadcasting can be computed in both a theoretical way and a numerical way. When we compute the E_{cost} in a theoretical way, we compute it as a ratio between a maximum broadcast rate and a sum of rates of all nodes. In order to get the maximum broadcast rate, we use a theoretically proved result stating that the maximum broadcast rate $C_{\min}(s)$ is the broadcast min-cut, and we compute the broadcast min-cut for a given graph with selected rates using the max flow algorithm in [79] over the hypergraph (see section 4.3.3 for details on the computing process). When we compute the E_{cost} in a numerical way, we compute it as a ratio between a generation size and a sum of transmission numbers of all nodes where both values are obtained from experimental simulation. There is a gap between the numerically computed E_{cost} and theoretically computed E_{cost}, but the gap becomes smaller as a generation size becomes larger as follows:

- $E_{cost} = \frac{\sum_{v \in N} C_v}{C_{\min}(s)} = \lim_{\text{generation size} \to \infty} \frac{\sum_{v \in N} \text{transmission number of node v}}{\text{generation size}}$
  - ◇ C_v: a rate of node v selected by a rate selection method
  - ◇ $C_{\min}(s)$: the achievable maximum broadcast rate with the selected rates.

The convergence of E_{cost} is also confirmed through simulation as seen in Figure 4.3 where Opt_{real} and Opt_{theory} represent the numerically computed E_{cost} and theoretically computed E_{cost} for an optimal solution respectively. The optimal solution is obtained by solving a linear optimization problem formulated in [13] for a given network. This convergence shows that theoretically computed E_{cost} could present a cost when network coding broadcasting with the evaluating algorithm is executed sufficiently long. In this chapter, we use both computations.

The cost from an optimal solution, $E_{optimal}$ is one direct reference for minimized cost that can be achieved with network coding. $E_{optimal}$ is obtained by solving a linear optimization problem with the following process: we first formulate a linear optimization problem of broadcasting for a given network and a given source as in [13]. Next, we solve the linear optimization problem using MOSEK, an optimization software for linear and convex

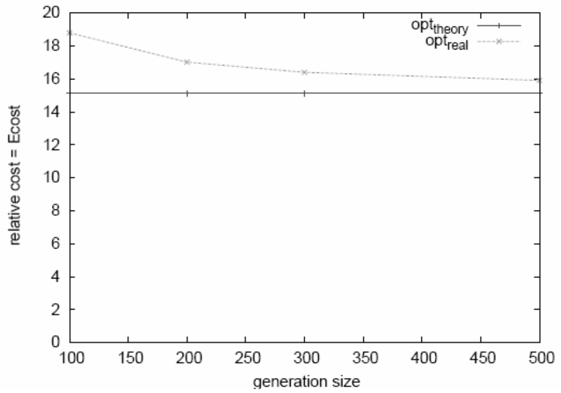


Figure 4.3: E_{cost} numerically and theoretically computed

optimization problems [80]. Then the sum of the rates in the optimal solution is $E_{optimal}$. Examples of optimal solutions for a sample random network and a lattice network are presented in Figure 4.4, where the radius of each circle represents a relative rate of each node. In Figure 4.4, an instance of a lattice graph contains a total node number $N = 21 \times 21$ and the neighbor number $M = 28$. An instance of a random unit disk graph contains the total node number $N = 200$ and the expected neighbor number $M = 20$.

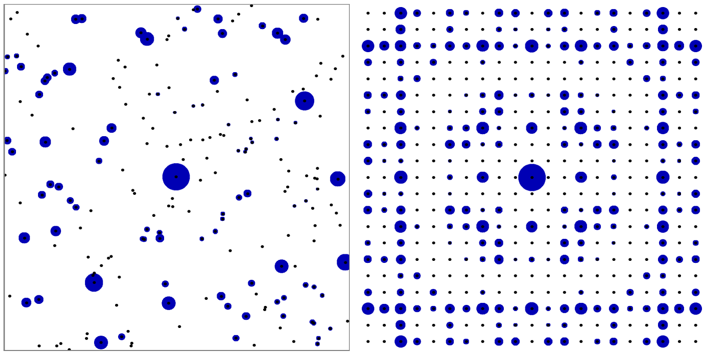


Figure 4.4: An example of optimal solution of a lattice graph and a random unit disk graph

As the other reference point, we use an indirect reference E_{bound}, which is the minimal transmission number when broadcasting achieves the maximal-efficiency-benefit of wireless network coding. Remember that the maximal-efficiency-benefit of wireless network coding is achieved when every single transmission can provide new and useful information to all its neighbors. Let us assume every transmission always reaches the maximum number of neighbors, M_{max}. Then, the transmission number to broadcast a data unit to N total nodes becomes at least $\frac{N}{M_{max}}$. Hence, E_{bound} becomes an indirect efficiency reference for the upperbound of efficiency.

4.3.2.2 Evaluation Metrics and Efficiency Without Network Coding

Here we provide three examples of reference without coding: an efficiency bound without coding $E^{(\text{no}-\text{coding})}_{\text{rel}-\text{cost}}$; E_{cost} of MPR-based broadcasting; and E_{cost} of CDS based broadcasting that forms CDS using a greedy algorithm with global topology.

An asymptotic efficiency bound without network coding, $E^{(\text{no}-\text{coding})}_{\text{rel}-\text{cost}}$ is obtained reusing the argument of [73]. Consider the broadcasting of one packet to a network and a node in the network which forwards the packets. To broadcast the packets to the whole network, another connected neighbor must receive the forwarded packets as seen in Figure 4.5. In a unit disk graph, these two connected neighbors share a neighborhood area, a gray area in Figure 4.5 and the size of the shared area is equal to $(\frac{2\pi}{3} - \frac{\sqrt{3}}{2})\rho^2 \approx 0.609$ where ρ is a transmission range. Any node inside the shared area receives duplicated packets and the duplicated receptions cause inefficiency. Considering the inefficiency, a relative cost for dense unit disk graphs, which is a ratio between the maximally covered area with one transmission and the maximally covered area without duplication, can be deduced into the following bound: $E^{(\text{no}-\text{coding})}_{\text{rel}-\text{cost}} \geq \frac{6\pi}{2\pi+3\sqrt{3}}$. Notice that $\frac{6\pi}{2\pi+3\sqrt{3}} \approx 1.6420\ldots > 1$.

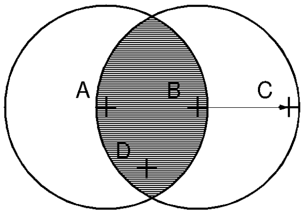


Figure 4.5: An efficiency bound without network coding

The other reference is E_{cost} of CDS based broadcasting that forms CDS using a greedy algorithm with global topology. We measured the cost with the transmission number from a source using CDS that is computed using a greedy algorithm [4] with total precise global topology. Broadcasting using the computed CDS is shown in Figure 4.6.

Similarly, E_{cost} of MPR-based broadcasting is also measured with the transmission number from the source to the whole network using MPR sets, and the MPR sets are computed with the algorithm of [6] described in section 2.1.

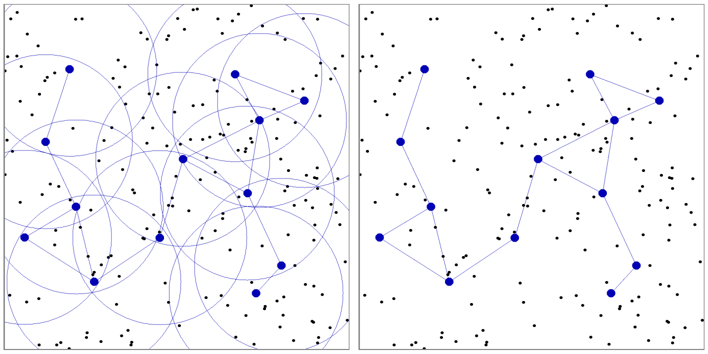


Figure 4.6: Connected Dominating Set

4.3.3 Efficiency Measuring Tools and Methods

In order to measure efficiency, we used three different methods:

- computing a maximum broadcast rate, i.e, the min-cut from a rate selection method for a given network
- simulating using the self-developed simulator with an idealized wireless network model
- simulating using NS-2 simulator (version 2.31)

In this section, we describe a method to compute a maximum broadcast rate and a self-developed simulator.

4.3.3.1 Computation of a Maximum Broadcast Rate over Hypergraph

For a given network, an achievable maximum broadcast rate obtained to use rate selection algorithms is theoretically computed as a broadcast min-cut. Hence, we compute the min-cut for the given graph as a theoretical performance of the evaluating algorithms. The min-cut computation is precisely done using a software library implementing the max-flow computation algorithm in [79] as follows. For a directed graph, we first computed max flows from a source to all nodes (theoretically a max flow is equal to the minimum capacity of all possible cuts from a source to a destination). Then, we take the minimum value among the computed max flows to each destination. This minimum value is equal to the minimum value among min-cuts from a source to each destination as defined (that is, a min-cut of broadcasting from a source to all destinations) in Equation 3.3 in section 3.3.2. In addition, we use optimizations for reusing search trees in [81] in order to speed up the computation of a max flow for each node.

Here, we encountered the problem that the min-cut computation is not done for a directed hypergraph but is done for a directed graph. Because our networks are wireless networks that are modeled by hypergrpahs, we must compute the min-cut of the directed hypergraph. Hence, we model a given network as a hypergraph, and then convert the hypergraph to a directed graph by inserting a virtual vertex to model physical layer broadcasting like [12], as seen in Figure 4.7. After converting the hypergraph, we assign an unlimited capacity the edges between the virtual vertex and real vertex. For example, we assign the unlimited capacity to an edge between $n1$ and $n1'$, and an edge between $n1'$ and $n2$. Finally we compute the min-cut using the converted directed graph when each node rate is selected by the evaluated algorithms.

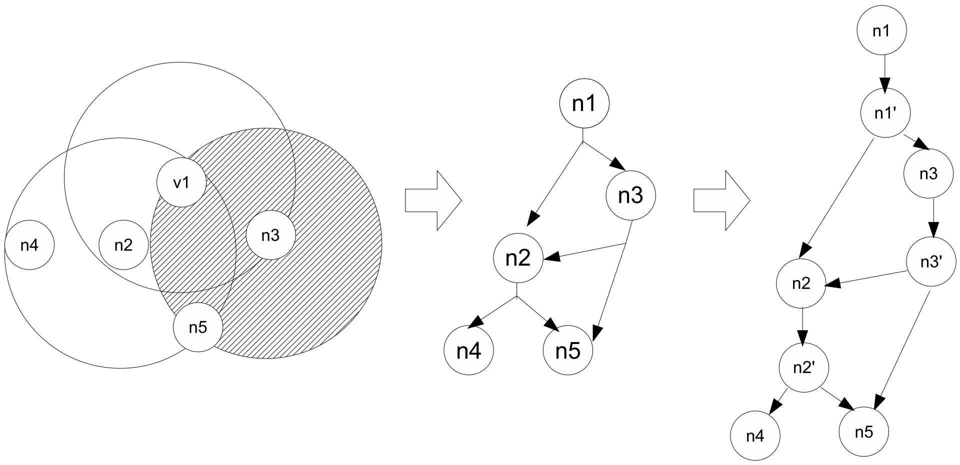


Figure 4.7: Conversion of a directed hypergraph for max flow computation

4.3.3.2 Simulation with Ideal Wireless Network Model

As the other way to measure the efficiency, we use the self-developed simulator with an ideal wireless model that has no contention, no collision and no delay (i.e. instant transmission and reception). With the ideal wireless model, all transmissions reach all nodes within a transmission range. In addition, our simulator assumes that nodes are distributed regularly in a lattice graph or are randomly distributed (their position is independent and identically distributed) in a random unit disk graph as defined in section 4.3.1. In addition, the transmission range of the nodes is decided by the expected number of neighbors M. For example, a transmission range can be $\rho = \frac{1}{\sqrt{\pi M}}$.

Execution of our self developed simulator is based on a discrete event scheduler. When our simulator schedules transmission, a packet transmission takes exactly one time unit and a node can either send or receive one packet at a time. When we simulate, random linear coding is used with various field sizes of $\mathbb{F}_p$ where p is 2, small(23), medium (17333) or large(1078071557).

4.4 Principal Rationale for Efficient Broadcasting with Network Coding

In this section, we introduce our principal rationale for efficient broadcast protocol with network coding. Before explaining our rationale, we first explain primitive operations to use random linear coding at each node in section 4.4.1. Then, we investigate the maximal-efficiency-benefit of wireless network coding in section 4.4.2 and explain our principal rational for efficient broadcasting with network coding in section 4.5.

4.4.1 Operations of each node for Broadcasting with Random Linear Coding

As explained in section 4.1, the main factor for efficient broadcasting with network coding is *rate selection*. When a rate of each node is decided by rate selection algorithms, basic operations for wireless broadcast protocols with random linear coding are almost the same as in Algorithm 2.

Algorithm 2: Random Linear Coding with Rate Selection

```
2.1 Source scheduling: the source transmits sequentially the $D$ vectors (packets) of a
    generation with rate $C_s$.
2.2 Nodes' start and stop conditions: The nodes start transmitting when they
    receive the first vector, and they continue transmitting until they and their
    neighbors have enough vectors to recover the $D$ source packets.
2.3 Nodes' scheduling: every node $v$ retransmits linear combinations of the vectors it
    has, and waits for a delay computed from the rate distribution.
2.4 Nodes' information storing condition : a node stores the received vector in its
    local buffer only if the received vector has new and different information from the
    information that the node already has.
```

In Algorithm 2, a source initiates network coding broadcasting by sending its original packets (vectors) at a rate C_s. Any node that receives the first vector starts transmitting a coded vector using random linear coding. For the transmission, a node stores the received vectors in its local buffer for a delay. When the node stores the received vectors, the node checks whether the received vectors have new (innovative) information or not, and stores only innovative vectors. After the delay, the node transmits the new coded packet that encodes all packets in its local buffer using random linear coding. The transmission continues until the node and its neighbors receive enough information to recover all D source packets.

With the basic operations of Algorithm 2, wireless broadcasting with network coding has only one parameter that varies. It is a delay that is computed as an approximation of a rate, when the rate is selected by rate selection algorithms. That is, when the rate of a node v at time t is selected as $C_v(t)$, the delay is computed as delay $\approx 1/C_v(t)$.

4.4.2 Basic Principle for Rate Selection

As explained in section 4.1, we can find an optimal transmission rate of each node by solving a linear optimization problem. However, the method does not necessarily provide direct insight about the optimal rates for wireless broadcasting with network coding and their associated optimal costs. Hence, we take another angle toward an efficient solution. The different approach is to specify the network coding protocol with the optimal rates based on some intuitive foresight and then compute the performance of the specified protocol, just as Fragouli et al starts with exhibiting an optimal algorithm for some regular networks in [73].

In order to get some intuitive foresight for the optimal rate, we first assume large networks. Considering the results of min-cut estimations for random graphs [62, 82–84], one intuition is that most nodes have similar neighborhoods; hence the performance with setting an identical rate for each node, deserves to be explored. This is the starting point of our rationale for efficient broadcasting with network coding.

To emphasize the insight regarding similar neighborhood size, we first assume a large lattice network with torus effect where nodes are regularly distributed and every node has the same number of neighbors, M. In addition, we also assume that the maximal-efficiency-benefit of wireless network coding is achieved so that every transmission brings *innovative* information to all receivers. Under this assumption, we can produce the logic of our basic principle:

1. Assume that the every node has an identical retransmission rate(IRON: Identical Rate for Other Nodes) as one, arbitrarily, for example one packet per second.
2. Then, every node with M neighbors can receive M coded packets per second. Assume that nearly all of them are innovative (non-redundant).
3. Then, the source should inject at least M packets per second.

With the above logic, we have the following heuristic:

- IRON (Identical Rate for Other Nodes): every node retransmits with rate one, except for the source, which transmits with a rate M.

Now, let us assume a large normal network where nodes are randomly distributed and there is no torus effect. In such networks, almost all nodes can receive M innovative packets per unit time with IRON only if almost all nodes have at least M neighbors. We explain how the receiving rate would be the expected number of neighbors M to all nodes that have more than M neighbors using an example of a random network where M is 5.

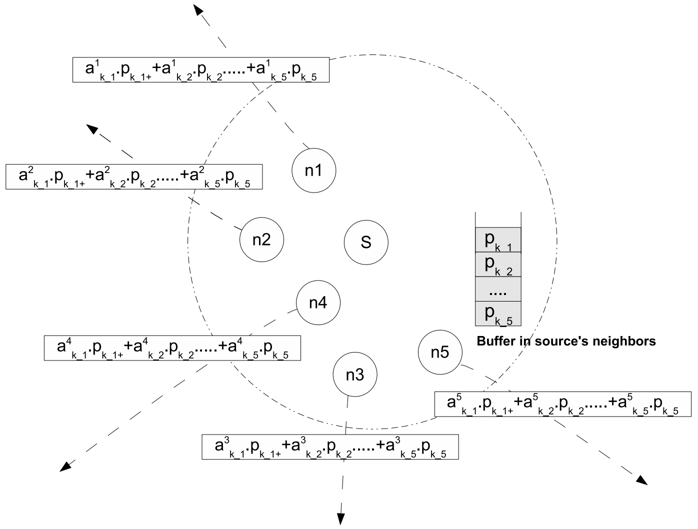

$a^1_{k\_1}.p_{k\_1+}+a^1_{k\_2}.p_{k\_2}.....+a^1_{k\_5}.p_{k\_5}$
$a^2_{k\_1}.p_{k\_1+}+a^2_{k\_2}.p_{k\_2}.....+a^2_{k\_5}.p_{k\_5}$
n1
n2
S
p_{k_1}
p_{k_2}
....
p_{k_5}
n4
Buffer in source's neighbors
$a^4_{k\_1}.p_{k\_1+}+a^4_{k\_2}.p_{k\_2}.....+a^4_{k\_5}.p_{k\_5}$
n5
n3
$a^5_{k\_1}.p_{k\_1+}+a^5_{k\_2}.p_{k\_2}.....+a^5_{k\_5}.p_{k\_5}$
$a^3_{k\_1}.p_{k\_1+}+a^3_{k\_2}.p_{k\_2}.....+a^3_{k\_5}.p_{k\_5}$

Figure 4.8: Retransmitting from source's neighbors with rate 1 with the source rate $M = 5$

With our basic heuristic, a source sends $M = 5$ packets per unit time and all others except the source transmit one coded packets per unit time. Notice that we assume an ideal network with no loss and collision. Hence, all neighbors of the source receive five source packets per unit time. Let us denote the source packets at kth unit time as $(P_{k_1}, P_{k_2}, P_{k_3}, P_{k_4}, P_{k_5})$. As seen in Algorithm 2, the first reception of an innovative packet lets the receivers start transmission. Hence the neighbors of the source transmit a result of random linear combination of the five source packets, as seen in Figure 4.8. In Figure 4.8, $a^i_{k_1}$ represents coefficients randomly chosen by a node i for P_{k_1}.

In order to achieve the maximal-efficiency-benefit of wireless network coding, we assume that all five coded packets from the source's neighbors are innovative to each other. Then nodes in $area2, area3$ and $area4$ receive two, three and three innovative encoded packets respectively from the transmission of the source's neighbors. The reception of the coded packets also lets nodes in $area2, area3$ and $area4$ transmit at rate one as seen in Figure

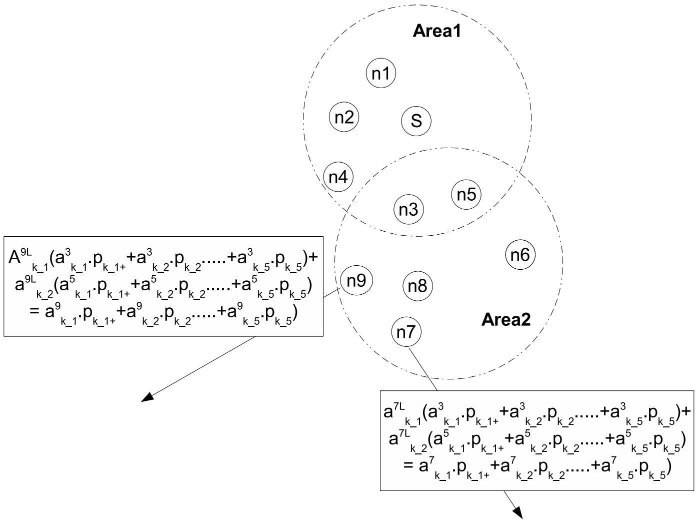

Area1
n1
n2
S
n4
n3
n5
n6
n9
n8
n7
Area2
$A^{9L}_{k\_1}(a^{3}_{k\_1}.p_{k\_1+}+a^{3}_{k\_2}.p_{k\_2}.....+a^{3}_{k\_5}.p_{k\_5})+$
$a^{9L}_{k\_2}(a^{5}_{k\_1}.p_{k\_1+}+a^{5}_{k\_2}.p_{k\_2}.....+a^{5}_{k\_5}.p_{k\_5})$
$= a^{9}_{k\_1}.p_{k\_1+}+a^{9}_{k\_2}.p_{k\_2}.....+a^{9}_{k\_5}.p_{k\_5})$
$a^{7L}_{k\_1}(a^{3}_{k\_1}.p_{k\_1+}+a^{3}_{k\_2}.p_{k\_2}.....+a^{3}_{k\_5}.p_{k\_5})+$
$a^{7L}_{k\_2}(a^{5}_{k\_1}.p_{k\_1+}+a^{5}_{k\_2}.p_{k\_2}.....+a^{5}_{k\_5}.p_{k\_5})$
$= a^{7}_{k\_1}.p_{k\_1+}+a^{7}_{k\_2}.p_{k\_2}.....+a^{7}_{k\_5}.p_{k\_5})$

Figure 4.9: Retransmitting from the source's two hop neighbors with rate 1

4.9. Notice that the innovative packet number from an area cannot exceed the number of innovative packets received. For example, all encoded packets from seven nodes in *area*4 cannot be innovative to each other because the nodes have only three innovative packets that are received from $n2$, $n3$ and $n4$ in *area*1. Hence, only three encoded packets among seven packets from the seven nodes in *area*4 are innovative to each other per unit time.

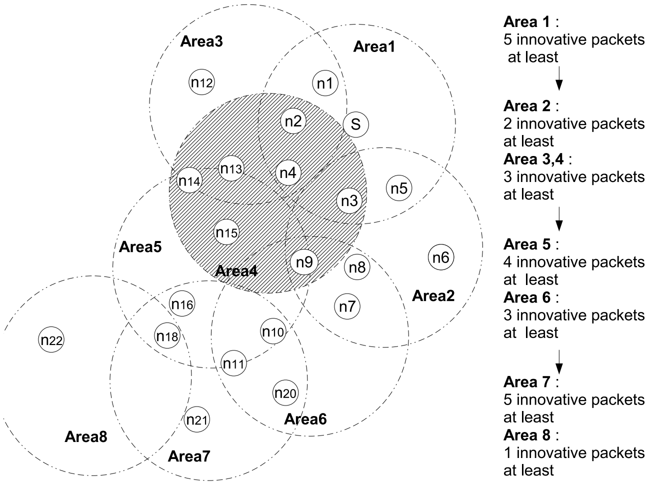


Figure 4.10: Innovative packet receptions at the nodes inside each area

With a similar process, the nodes in *area*5 and *area*6 receive the transmitted coded packets from nodes in *area*2, *area*3 and *area*4 as seen in Figure 4.10. Although the transmitted packets cannot be all innovative any more, we assume the best case, which is that nodes receive the maximum number of innovative packets from the transmitting area. For example, nodes in $area6$ receive two innovative packets from $area2$, and nodes in $area5$ receive three innovative packets from $area4$. Now, the nodes in $area5$ and $area6$ also generate coded packets and transmit them, and the nodes in $area7$ receive them. From this point, the number of possible innovative packets from $area7$ can be five because three innovative packets arrived through the path $area1 \rightarrow area3 \rightarrow area4 \rightarrow area5 \rightarrow area7$ and two innovative packets arrived through the path $area1 \rightarrow area2 \rightarrow area6 \rightarrow area7$. Therefore, the possible number of innovative packets becomes $M = 5$. If every node continues transmission in this way, the receiving rates of the nodes in $area7$ become $M = 5$, the intended receiving rate in order to achieve the maximal-efficiency-benefit of wireless network coding.

This example shows that with IRON nodes only three hops away from the source can have the intended receiving rate $M = 5$ for the maximal-efficiency-benefit of wireless network coding. If we generalize the example, we see that nodes located away from the source can have the intended receiving rate M for the maximal-efficiency-benefit of wireless network coding with high probability. This means almost all nodes in large networks could have the intended receiving rate for the maximal-efficiency-benefit of wireless network coding.

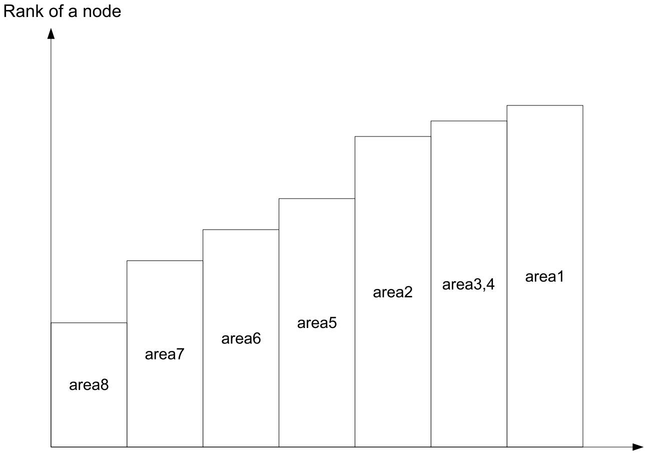
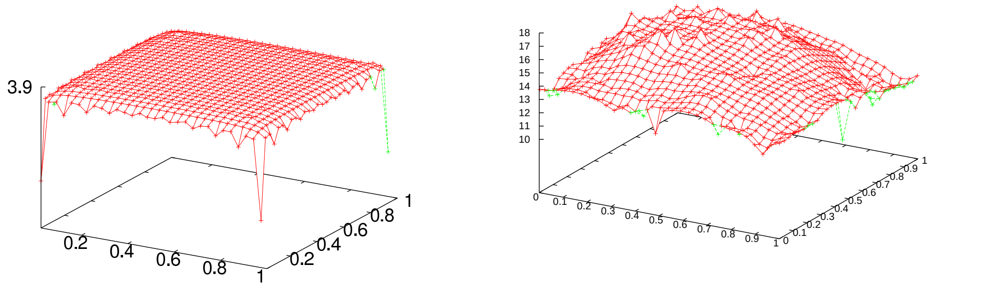


Figure 4.11: A gap of rank of nodes

Then, we can ask whether nodes located closer to the source could not have an intended rate for the maximal-efficiency-benefit of wireless network coding. To find the answer to this question, we must notice that we have the explained situation only for some initial period after broadcasting with network coding starts. If nodes located closer to the source continue to receive fewer innovative packets than $M = 5$ like the example, a gap is created between the amount of accumulated innovative information inside each node, and the gap grows as seen in Figure 4.11. The growing gap increases the possible innovative packet number that nodes in an area can send. For example, before the gap grows, the nodes in $area6$ can receive only two innovative packets from $area2$. But, after this gap grows, the nodes in $area6$ definitely can receive three innovative packets from all common nodes $(n7, n8, n9)$ between $area6$ and $area2$. Similarly, nodes in $area6$ also can receive three innovative packets from all common nodes $(n9, n10, n11)$in $area4$. As a consequence, the nodes in $area6$ can receive $M = 5$ innovative packets per unit time.

With the same reasoning for all areas, we can see that *with IRON, most nodes can have the intended receiving rate for the maximal-efficiency-benefit of wireless network coding after some initial period*. The statement is still not for all nodes, but for most nodes because of an exceptional case. The exceptional case is $n22$ in $area8$, whose neighborhood is remarkably small. Because $n22$ has only one neighbor $n18$ and the transmission rate of $n18$ is just one, the receiving rate of $n22$ cannot exceed one even if $n18$ has more innovative information to send. From this observation, we can see that *with IRON a node cannot have the intended receiving rate to achieve the maximal-efficiency-benefit of wireless network coding if a node has a smaller neighborhood than the intended rate (M).*



(a) min-cut with IRON on a lattice

(b) min-cut with IRON on a random unit disk

Figure 4.12: Min-cut with IRON heuristic

In order to illustrate that the smaller neighborhood causes the smaller receiving rate (a min-cut) of IRON, we drew a graph showing the relationship between each node's position and its receiving rate (a min-cut) in Figure 4.12. The graphs in Figure 4.12 were obtained by computing the min-cut in a lattice graph and a random unit disk graph. Figure 4.12(a) represents a result of a 20×20 lattice graph where every node has exactly 4 neighbors except nodes around the border, and the source is located in the middle of the network. Figure 4.12(b) represents the results of a random unit disk graph where there exist 400 total nodes with a 20 node density (the average number of nodes in a range= 20), and the source is the node with the largest neighborhood. In both graphs, both the x and y axis represent the position of each node on the plane; the value on the vertical z axis represents a receiving rate, a max flow and a capacity of min-cut. For the randomly generated unit disk graph, there is interpolation. Note that the min-cut presented in Figure 4.12 is the max flow (a capacity of the min-cut) between a source and each destination; it is not a max flow from the source to all nodes, (or the maximum broadcast rate). The maximum broadcast flow (rate) is the minimum of capacities of all min-cuts between the source and all nodes as stated in section 3.3.2. Each max flow (a capacity of min-cut) is obtained by using a software library implementing the max-flow computation algorithm [79] after converting a hypergraph modeling a wireless network into a general graph as explained in section 4.3.3.

Most of all, the graph in Figure 4.12(a) clearly shows that only nodes around the border have a lower min-cut than other nodes (the min-cut at the border is 2). The result confirms our intuition that a small neighborhood causes a low receiving rate (low capacities of min-cuts and low max flows), and the low capacities of min-cuts of a few nodes finally bound a maximum broadcast rate at 2 instead of 4. Similarly, results in Figure 4.12(b) show nodes around the border have low min-cuts, but some nodes away from the border also have low min-cuts like $n22$ in Figure 4.11. Hence, for general networks, we introduce an adapted rate selection algorithm based on local topology information.

4.4.3 Heuristic for Rate Selection: IRMS

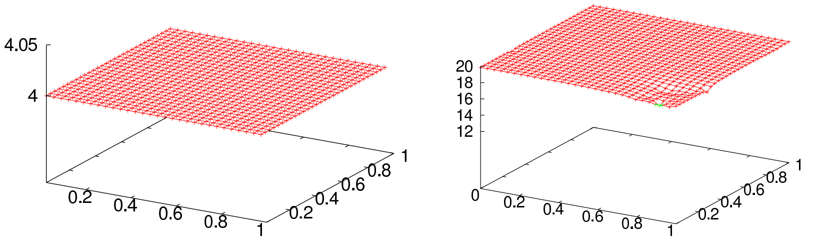
As illustrated in Figure 4.12, a small neighborhood may cause a node to have a smaller min-cut than the intended value, M, and the small neighborhood consequently bounds the source's transmitting rate. We call the problematic nodes with a smaller receiving rate (a min-cut) *"starving nodes"*.

In order to alleviate the bottleneck from the starving nodes, their neighbors compensate for the starvation by increasing their transmission rates. For example, $n18$ in Figure 4.10increases its rate to 5 in order to let the starving node $n22$ have the intended receiving rate (the local min-cut) $M = 5$. Hence, we have the following heuristic;

- IRMS (Increased Rate for the Most Starving node): the rate of a node v is set to C_v, with:

$C_v = \frac{C_s}{min_{u \in H_v}(|H_u|)}$, where H_w is the set of neighbors of w, and C_s is the source rate which is M.

This rate selection, IRMS adjusts step 2 of the basic reasoning for IRON, to accommodate for the fact that some nodes have fewer than the expected number of neighbors M. The adjustment uses a rate selection rule to help the starving nodes according to how much they starve by generalizing the observed intuition. In other words, a node monitors how much its neighbors starve and increases its rate until the most starving neighbor has a sufficient receiving rate. Precisely, if a node v has less than $|H_v| < M$ neighbors, it would still receive a packet rate $\geq \frac{M}{|H_v|}$ from each of its $|H_v|$ neighbors. Hence, an overall rate at least equal to M. Notice that a min-cut is not necessarily larger than M, $C_{\min}(s, v) \geq M$ because a receiving rate at every node is at least M and it is generally larger than M. The reason that a min-cut cannot exceed M is that a min-cut cannot be larger than a source rate and the source rate is M.



(a) min-cut with IR-MS, on a lattice (b) IR-MS, on a random unit disk

Figure 4.13: Min-cut with IRMS

In order to illustrate the improvement of a min-cut for each destination with IRMS, we use a graph similar to Figure 4.12 using the same parameters except for the rate selection algorithm. As shown in two graphs of Figure 4.21, almost all nodes have the targeted min-cut M, which is 4 in a lattice graph and 20 for a random unit disk graph. This means that IRMS solves the bottleneck to bound the maximum broadcast rate. The increased rates of IRMS may bring additional costs, but this additional cost is comparatively much lower than the increase of min-cut. Hence, efficiency of broadcasting with IRMS is overall improved. We present more systematic experiments on the performance of the heuristic IRMS in section 4.5.

Notice that [73] also proposed the same expression of C_v, except that it was multiplied by a coefficient k. It was in a slightly different context because their work was for all-to-all broadcasting. For one-to-all broadcasting, our theoretical argument of section 4.5 and of [85] implied that $k = 1$ was sufficient. In our algorithm, eventual decoding at every node can be ensured by a termination phase where a node continues transmission until all its neighbors can recover all original data from a source, and the length of the termination phase is negligible for large generations. For one-to-all broadcasting, IRMS would have a threefold gain compared to the theoretical arguments of [73] by selecting $k = 1$ instead of $k = 3$. In fact, some intensive experiments over a random unit disk graph detailed in section 4.6 showed that the actual min-cut would be sometimes less than the target min-cut= M. Hence, the gain would be roughly 2 if we use the heuristic in [73] for one-to-all communication.

4.5 Theoretical Analysis of Efficiency

In this section, we theoretically study the efficiency of our approach in one dimensional torus networks in section 4.5.1, and next in two dimensional lattice networks and random networks on Euclidean spaces in section 4.5.2. The theoretical study focuses on theoretically computing $E_{rel-cost}$ that presents a ratio between E_{cost} of our approach and E_{bound}. E_{cost} is a transmission number to broadcast one unit data to the whole network, and is theoretically computed as a ratio between a sum of rates selected by our approach and the achievable broadcast rate with selected rates as stated in section 4.3.2. E_{bound} is an efficiency upper bound for the maximal-efficiency-benefit of wireless network coding as detailed in section 4.3.2. Hence, we can say our approach is optimal if $E_{rel-cost}$, a ratio of the computed E_{cost} and E_{bound} converges to one. In this section, by showing that the ratio theoretically converges to one in one dimensional torus networks and in two dimensional networks, we show the optimality of our approach.

One essential part to show the optimality is the theoretical computation of E_{cost} of our approach. The computation should use a theoretical maximum broadcast rate of our approach, and the theoretical maximum broadcast rate is a min-cut of broadcasting as detailed in section 3.3. The goal of our approach is that the min-cut should be the expected number of neighbors M in order to achieve the maximal-efficiency-benefit of wireless network coding with low cost. Hence, we prove that our approach can achieve the intended min-cut M, and then use the proved result in order to show the optimality of our approach.

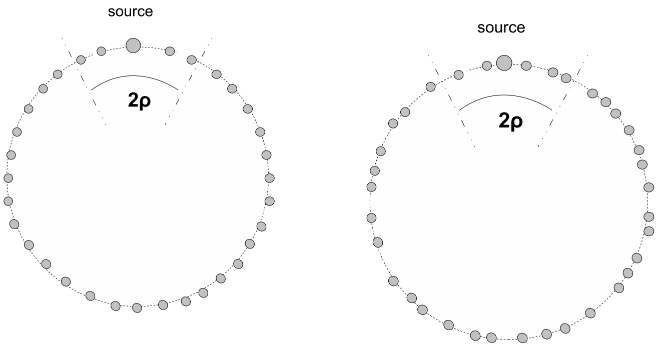
This section is precisely organized as follows: section 4.5.1 proves the asymptotical optimality of our approach using IRON in one dimensional torus networks and section 4.5.2 proves the asymptotical optimality using IREN/IRON, which an is extension of IRON for two dimensional networks on Euclidean spaces.

4.5.1 Theoretical Analysis of Efficiency in 1D lattice torus networks

In this section, we prove the optimal efficiency of our approach in one dimensional torus networks. From the proof, we deduce that the efficiency of our approach is asymptotically optimal, and two times higher than the efficiency of a Connected Dominating Set in one dimensional torus networks. For the proof, we first detail the network model, then explain theoretically achieved efficiency, and lastly prove that a maximum broadcast rate (a max flow and a min-cut) of our approach is the expected number of neighbors M as intended.

4.5.1.1 Network Model

We assume a line network of length L with torus effect. The network is a loop as seen in Figure 4.14; Figure 4.14(a) represents a one dimensional torus lattice network and Figure 4.14(b) represents a one dimensional torus random network.



(a) 1D torus lattice network (b) 1D torus random network

Figure 4.14: an example of 1D torus network topology

Let $\mathcal{V}$ denote a set of nodes in the network. The nodes are distributed with the same interval in a one dimensional lattice torus network as seen in Figure 4.14(a) and they are randomly distributed (independent and identically-distributed) in a one dimensional random torus network as seen in Figure 4.14(b). Among the nodes, two nodes become neighbors if the distance along the loop between two nodes is smaller than a transmission range ρ. Namely, the neighborhood R of a node includes nodes within a distance of ρ at the right side and left side. Given a one dimensional torus network, we define the *positive half-neighborhood* R_+ (resp. *negative half-neighborhood*, R_-), as the set of neighbors of v after v in a counter-clockwise direction (resp. clockwise direction).

Let N denote the total number of nodes and M denote the expected number of neighbors. Then, the number of nodes in the neighborhood R is exactly M in a lattice graph, and asymptotically M in a random graph. The expected neighbor number M is also represented using the density μ and transmission range ρ as $M \propto \mu\rho$ (in lattice, $\mu = 1$).

4.5.1.2 Near Optimality

In order to evaluate the efficiency of our approach, IRON, we compute $E_{rel-cost}$ using E_{bound} and theoretical E_{cost} of our approach: $E_{rel-cost} = \frac{E_{cost}}{E_{bound}} \geq 1$. As $E_{rel-cost}$ approaches one, our approach becomes optimal. E_{bound} offers a standard to measure the optimality of energy-efficiency as explained in section 4.3.2.1, and is computed as $\frac{N}{M_{max}}$, which is the minimum transmission number to broadcast a data unit to N total nodes when M_{max} is the maximum number of neighbors. E_{cost} is the transmission number to deliver one unit data. Because our method, IRON assigns the expected number of neighbors M to a rate of the source and one to a rate of other nodes and C_{min}, as minimum network cut, is the maximum broadcast rate of IRON, E_{cost} of IRON is computed as $\frac{N-1+M}{C_{min}}$. By showing that $E_{\text{rel-cost}}$, the result of a comparison between these two values (E_{cost} and E_{bound}) converges to one, we will prove the optimal efficiency of our method, IRON. For the proof, we show the convergence in one dimensional lattice torus networks and in one dimensional random torus networks.

We first prove the convergence of $E_{\text{rel-cost}}$ to one in one dimensional lattice torus networks. By definition, we have $N = \mu L$ and $M = 2\rho\mu$. In addition, $\mu = 1$, $M_{max} = M$ and ρ is fixed in lattice one dimensional torus networks, and hence, with Theorem 2 $C_{min} = M$, we have :

$$E_{rel-cost} = \frac{E_{cost}}{\frac{N}{M_{max}}} = \frac{N-1+M}{N}\frac{M_{max}}{C_{min}} = 1 + O(\frac{1}{L}) \tag{4.1}$$

Hence, $E_{rel-cost}$ converges to one as networks become larger ($L \to \infty$).

Second, we prove the convergence of $E_{\text{rel-cost}}$ to one in one dimensional random torus networks. By definition, we have $N = \mu L$, $M = 2\rho\mu$, and the fixed ρ. In addition, with Theorem 3, $\frac{M_{max}}{M} \xrightarrow{p} 1$ and $\frac{M}{C_{min}} \xrightarrow{p} 1$ as networks becomes larger and denser. Hence we have :

$$E_{rel-cost} = \frac{E_{cost}}{\frac{N}{M_{max}}} = \frac{N-1+M}{N}\frac{M_{max}}{M}\frac{M}{C_{min}} \xrightarrow{p} (1 + O(\frac{1}{L})) \tag{4.2}$$

Hence, $E_{rel-cost}$ converges to one as networks become larger ($L \to \infty$).

For both one dimensional lattice torus networks and one dimensional random torus networks, $E_{rel-cost}$ converges to one. Therefore, we prove that network coding with our heuristic IRON is asymptotically optimal in one dimensional torus networks.

In addition, E_{cost} from connected dominating set in one dimensional lattice torus networks is $\frac{N}{\frac{M}{2}} = \frac{2N}{M}$ considering the transmission range ρ is overlapped to connect the nodes in a dominating set. Therefore, our heuristic IRON is two times more efficient than a Connected Dominating Set in one dimensional lattice torus networks. In addition, in one dimensional dense random networks, the number of nodes in a transmission range converges to M, and hence E_{cost} from connected dominating sets converges to $\frac{N}{\frac{M}{2}} = \frac{2N}{M}$. Therefore, our heuristic IRON is asymptotically two times more efficient than a Connected Dominating Set in one dimensional random torus networks as the networks become dense.

4.5.1.3 Proof of the Achievable Broadcast Rate in one dimensional lattice torus networks

In this section, we prove that an achievable broadcast rate of network coding with our approach is the expected number of neighbors M in one dimensional lattice torus networks. Note that the maximum broadcast rate is the minimum value of a capacity of all possible cuts to all nodes. As defined in Equation 3.4 in section 3.3.2: $\Delta S \triangleq \{v \in S : \mathcal{N}(v) \cap T \neq \emptyset\}$ and $C(S) \triangleq \sum_{v \in \Delta S} C_v$, a capacity of a cut is the sum of rates of nodes in ΔS.

From Equation 3.4, we have the following lemma, and from Lemma 1, we have Theorem 1

Lemma 1 $C(S) \geq |\Delta S|$

Proof: Because C_s (the source's rate) is $M \geq 1$ and C_v (other node v's rate $v \in (V \setminus \{s\})$) is one with IRON, the sum of rates of node v in ΔS is at least equal to the size of ΔS.

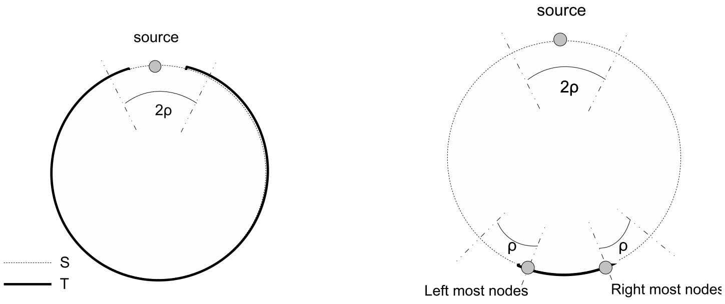
Theorem 1 $C(S) \geq M$

Proof: With Lemma 1, $|\Delta S|$ decides $C(S)$. ΔS is a union of nodes in R_- of the leftmost posed node in all possible Ts ($R_-^{leftmost}$)and nodes in R_+ of the rightmost posed node in the possible Ts ($R_+^{rightmost}$) as seen in Figure 4.15(b). When $s \in \Delta S$ as seen in Figure 4.15(a), neighbors of the source s exist in T, and hence $C(S) \geq M$. When $s \notin \Delta S$, $|\Delta S| = |R_-^{leftmost}|+|R_+^{rightmost}|$. In this case, all nodes in ΔS have the identical transmission rate as one. Hence, the sum of rates of nodes in ΔS is equal to the size of ΔS, and $|\Delta S| = |R_-| + |R_+|$. In one dimensional lattice torus networks, every node has the same number of half neighbors ($|R_-| = |R_+| = \frac{M}{2}$). This means $|R_-^{leftmost}| + |R_+^{rightmost}| = M$. Therefore, $|\Delta S| = M$, and $C(S) = M$.

The result of Theorem 1 results in a property of the capacity of every s-t min-cut.

Theorem 2

$$\text{For any } t \in (T\backslash\{s\}), C_{min}(s,t) = M \text{ and as a result, } C_{min}(s) = M \tag{4.3}$$



(a) $s \in \Delta S$ (b) $s \notin \Delta S$

Figure 4.15: S/T partition in a one dimensional torus network

Proof: Let S_{min}/T_{min} be a cut with minimum capacity: $C(S_{min}) = C_{min}(s,t)$. With Theorem 1, $C_{min}(s,t) \geq M$. Conversely, let us consider a specific cut, $S_s = \{s\}$ and $T_s = \mathcal{V} \setminus \{s\}$. Then, the source s obviously has a least one neighbor that has to be in T, hence $\Delta S = \{s\}$. The capacity of the cut is $C(S_s) = \sum_{v \in \Delta S} C_v = C_s = M$, and thus $C_{min}(s,t) \leq M$. Therefore, $C_{min}(s) = M$.

4.5.1.4 Proof of the Achievable Broadcast Rate in one dimensional random torus networks

In this section, we prove that a ratio between the achievable broadcast rate C_{min} and the expected number of neighbors (M) converges to one as networks become denser and larger ($\frac{M}{C_{min}} \xrightarrow{p} 1$). In addition, we also prove that the ratio between the expected number of neighbors (M) and the maximum number of neighbors M_{max} converges to one as networks become denser and larger ($\frac{M_{max}}{M} \xrightarrow{p} 1$).

With similar logic shown in the proof of Theorem 2 with Lemma 1 and Theorem 1 in one dimensional lattice networks, we prove the following theorem in one dimensional random networks.

Theorem 3

$$\frac{M}{C_{min}(s)} \xrightarrow{p} 1 \tag{4.4}$$

Also,

$$\frac{M_{max}}{M} \xrightarrow{p} 1 \tag{4.5}$$

Proof: As seen in Theorem 2, $C_{min}(s,t)$ decides $C_{min}(s)$ and $C_{min}(s,t)$ is decided by $C(S)$. With Lemma 1, $C(S)$ is decided by $|\Delta S|$. Hence, minimum $|\Delta S| = C_{min}(s)$.

Because $|\Delta S| = |R_-| + |R_+|$, minimum $|\Delta S|$ = minimum $|R_-|$+ minimum $|R_+|$.

When a position of a node is i.i.d , for all i $\in \{1, .., N\}$, $|R_+|$ is $X = Y_1 + Y_2 + ...Y_{N-1}$, where Y_i is i.i.d Bernoulli trials of probability p, and $p = \frac{M}{2}\frac{1}{N}$ is probability for one node to be in R_+ of a given node. ($0 \leq p \leq 1$, N: total number of nodes, M: average number of neighbors).

Let X be the number of neighbors of a given node. Because X is a random variable, $P(X < a) = P(e^{-\lambda X} > e^{-\lambda a}) < \frac{E(e^{-\lambda X})}{e^{-\lambda a}}, \forall \lambda > 0$ with the Chernov bound in [86].

$E(e^{-\lambda X}) = \sum_{k=0}^{N-1} P(X = k)e^{-\lambda k} = \sum_{k=0}^{N-1} \binom{n-1}{k} p^k(1-p)^{N-1-k}e^{-\lambda k} = (1 - p + pe^{-\lambda})^{N-1}$.

With $a = Np(1-\delta)$, $P(X < Np(1-\delta)) < \frac{E(e^{-\lambda X})}{e^{-\lambda a}} = \frac{(1-p+pe^{-\lambda})^{N-1}}{e^{-\lambda Np(1-\delta)}}$.

With $p = \frac{M}{2}\frac{1}{N}$, $P(X < \frac{M}{2}(1-\delta)) < (1 + \frac{M}{2}\frac{1}{N}(e^{-\lambda} - 1))^{-1}e^{N\ln(1+\frac{M}{2}\frac{1}{N}(e^{-\lambda}-1))}e^{\lambda(1-\delta)\frac{M}{2}}$.

Using $\ln(1-x) < -x$, $P(X < \frac{M}{2}(1-\delta)) < (1+\frac{M}{2}\frac{1}{N}(e^{-\lambda}-1))^{-1}e^{(N-1)\frac{M}{2}\frac{1}{N}(e^{-\lambda}-1)}e^{\lambda(1-\delta)\frac{M}{2}}$.

By fixing $\lambda = \delta$, $P(X < \frac{M}{2}(1-\delta)) < (1 + \frac{M}{2}\frac{1}{N}(e^{-\delta} - 1))^{-1}e^{\frac{M}{2}(e^{-\delta}-1+\delta-\delta^2)}$.

Using $e^{-\delta} < 1 - \delta + \frac{\delta^2}{2}$, $P(X < \frac{M}{2}(1-\delta)) < (1 + \frac{M}{2}\frac{1}{N}(e^{-\delta} - 1))^{-1}e^{-\frac{M}{4}\delta^2}$.

Then, finally we have $P(X < \frac{M}{2}(1-\delta)) < (1 - \frac{M}{N})^{-1}e^{-\frac{M}{4}\delta^2}$.

When $N \to \infty$ and $M \to \infty$ with $\delta > 0$, $(1 - \frac{M}{N})^{-1}e^{-\frac{M}{4}\delta^2} \to 0$. With the same logic, we have the same bound for $|R_-|$. Therefore $\frac{M}{C_{min}(s)} = \frac{M}{minimum|R_+|+minimum|R_-|} \xrightarrow{p} 1$.

In addition, $P(X > a) = P(e^{\lambda X} > e^{\lambda a}) < \frac{E(e^{\lambda X})}{e^{\lambda a}}, \forall \lambda > 0$ with the Chernov bound in [86].

$E(e^{\lambda X}) = \sum_{k=0}^{N-1} P(X = k)e^{\lambda k} = \sum_{k=0}^{N-1} \binom{n-1}{k} p^k(1-p)^{N-1-k}e^{\lambda k} = (1 - p + pe^{\lambda})^{N-1}$.

With $a = Np(1+\delta)$, $P(X > Np(1+\delta)) < \frac{E(e^{\lambda X})}{e^{\lambda a}} = \frac{(1-p+pe^{\lambda})^{N-1}}{e^{\lambda Np(1+\delta)}}$.

With $p = \frac{M}{2}\frac{1}{N}$, $P(X > \frac{M}{2}(1+\delta)) < (1 + \frac{M}{2}\frac{1}{N}(e^{\lambda} - 1))^{-1}e^{N\ln(1+\frac{M}{2}\frac{1}{N}(e^{\lambda}-1))}e^{-\lambda(1+\delta)\frac{M}{2}}$.

Using $\ln(1-x) < -x$, $P(X > \frac{M}{2}(1+\delta)) < (1+\frac{M}{2}\frac{1}{N}(e^{\lambda}-1))^{-1}e^{(N-1)\frac{M}{2}\frac{1}{N}(e^{\lambda}-1)}e^{-\lambda(1+\delta)\frac{M}{2}}$.

By fixing $\lambda = \ln(1+\delta)$, $P(X > \frac{M}{2}(1+\delta)) < (1 + \frac{M}{2}\frac{\delta}{N})^{-1}e^{\frac{M}{2}(\delta-(1+\delta)\ln(1+\delta))}$.

Then, we have $P(X > \frac{M}{2}(1+\delta)) < (1 + \frac{M}{2}\frac{\delta}{N})^{-1}e^{-\frac{M}{4}(\delta^2+O(\delta^3))}$, and finally $P(X > \frac{M}{2}(1+\delta)) < (1 - \frac{M}{N})^{-1}e^{-\frac{M}{4}\delta^2}$.

When $N \to \infty$ and $M \to \infty$ with $\delta > 0$, $(1 - \frac{M}{N})^{-1}e^{-\frac{M}{4}\delta^2} \to 0$. With the same logic, we have the same bound for $|R_-|$. Therefore $\frac{M_{max}}{M} = \frac{maximum|R_+|+maximum|R_-|}{M} \xrightarrow{p} 1$.

4.5.2 Proof of Asymptotical Optimality in 2 dimensional Euclidean Space

In this section, we prove the optimal efficiency of our approach in two dimensional networks on Euclidean spaces. Just as we did in our proof for one dimensional torus networks, we prove the optimal efficiency by showing that a ratio ($E_{rel-cost}$) between E_{cost} of our approach and E_{bound} converges to one. From here, a network in this section (section 4.5.2) implies a two dimensional network on Euclidean spaces.

For $E_{rel-cost}$ computation, we theoretically compute E_{cost} as a ratio between a sum of rates selected by our approach and an achievable broadcast rate (a max flow and a min-cut for broadcasting) with the selected rates. As proved in one dimensional torus networks, an achievable broadcast rate of our approach is also proved to be the expected number of neighbors M both in lattice networks and random networks. For the achievable broadcast rate proof, we extend the basic rate selection IRON relying on the fact that networks become regular as they become larger and denser. Because of the regularity, most nodes have the similar neighborhood size except for the nodes around the border. As we explained in section 4.4, the nodes with smaller neighborhoods than other nodes decrease an achievable broadcast rate. Hence, the extended rate selection must assign different rates to the exceptional nodes around the border. Precisely, the extended rate selection assigns the intended receiving rate (a max flow and a min-cut) M to the exceptional nodes and their neighbors in order to safely achieve the intended maximum broadcast rate for the maximal-efficiency-benefit of wireless network coding. We detail the extended rate selection, IREN/IRON in section 4.5.2.

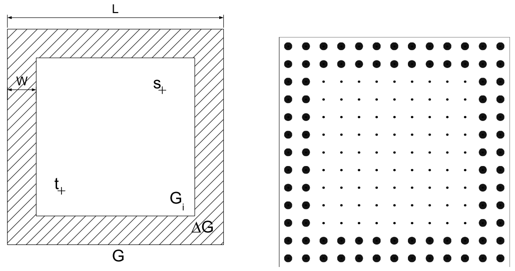
One essential step to show optimality is to prove that the extended rate selection, IREN/IRON can achieve the intended maximum broadcast rate M for the maximal-efficiency-benefit of wireless network coding. The proof proceeds as follows: we first link a capacity of a cut between nodes in S and nodes in T with the number of nodes in S that are neighbors of nodes in T. The number of these nodes decides the capacity of the cut, as defined in Equation 3.4 in section 3.3.2 except for the case where the source is included in such nodes (ΔS in equation 3.4). To compute the number of these nodes, we use the fact that the neighborhood in two dimensional Euclidean space is obtained with a Minkowski sum; a Minkowski sum is a classical way to express the neighborhood of an area. In other words, we compute the number of nodes deciding a capacity of a cut by computing the neighbor number with the inequality on Minkowski sums. However, because of the effect from the border, several special cases exist for applying the inequality. Hence, we classify the several cases, and for each category we prove that the capacity of the cut has the desired bound, M.

This section is organized as follows: section 4.5.2.1 presents additional definitions for explaining the principle and the proof; section 4.5.2.2 details the extended rate selection, IREN/IRON; section 4.5.2.5 presents the proof of the asymptotical optimality of IREN/IRON in lattice networks and in random networks, assuming that the achievable maximum broadcast rate is M; and finally section 4.5.3 presents the proof of the achievable maximum broadcast rate.

4.5.2.1 Additions to Network Model and Definitions

additional definition

We consider a network inside a square area such as the one in Figure 4.16(a) and let L denote the edge length of the square area G containing the network.



(a) A sample square network

(b) selected rate of IREN/IRON

Figure 4.16: a sample network inside a square

- The radio transmission range from a node inside a network is ρ.
- For a lattice, we denote R the set of neighbors of the origin node $(0, 0)$, as represented in Figure 4.2(a): $R \triangleq \{(x, y) \in \mathbb{Z}^2 : x^2 + y^2 \leq \rho^2\}$
- M denotes the "expected" neighbor number of one node. For a lattice graph, the number is: $M = |R| - 1$, and for a random unit disk graph with N nodes, the number is $M = \pi\rho^2\mu = \pi\rho^2\frac{N}{L^2}$ on the condition that the node exists inside the interior area G_i defined in the following list.

We define the *border area* as the area of the fixed width $W > \rho$ from each edge of the square network G, and *border nodes* as the nodes inside the border that area. Hence, the area $L \times L$ of G is partitioned into:

- ΔG, the border, with area $A_{\Delta G} = 4W(L-W)$ in Figure 4.16, the hatched area ΔG
- G_i, the "interior" $G_i \triangleq G \setminus \Delta G$, with area $A_{G_i} = (L-2W)^2$

For a lattice graph, we let Γ be a full (with full rank) , unbounded, *integer (point) lattice* in n-dimensional space; Γ is the set $\mathbb{Z}^n$, where the lattice points are n-tuples of integers. For the Γ, $\mathcal{L}, \mathcal{L}_i, \Delta\mathcal{L}$ are used for each part of the above definitions for the square network G. Formally for a full, unbounded, *integer lattice* lattice Γ :

- $\mathcal{L} = \Gamma \cap G$
- $\mathcal{L}_i = \Gamma \cap G_i$
- $\Delta\mathcal{L} = \Gamma \cap \Delta G$

As described in the first list, M denotes the expected number of neighbors of one node. For a node in the interior area of a lattice network, every node has the same number of neighbors and hence the expected number of neighbors is exactly the number of points in the neighborhood R minus one (see Figure 4.2(a) and Figure 4.2(b)) : $M = |R| - 1$. For a random disk unit graph with N nodes, M is related to the network density μ and the radio transmission range. If a node exists inside an interior area, M is : $M = \pi\rho^2\mu$, and if a node exist inside a border area, M is $\frac{\pi\rho^2\mu(R\cap\Delta G)}{R}$. The network density μ is deduced from the size of network edge and the total number of nodes : $\mu = \frac{N}{L^2}$. The expected number of neighbors is used to prove the achievable maximum broadcast rate for a random unit disk graph.

The interior area G_i has one requirement that all nodes inside the interior area G_i are out of range of the outside of the network. The requirement for G_i is satisfied by selecting sufficiently large W. For a random unit disk graph the value of W, for instance, should be the transmission range ρ: $W = \rho$. For a lattice graph, R should be included in the disk of radius ρ such as $W \le \rho$. The requirement on W is combined with the requirement for R in section 4.3.1. The requirement also leads a property for G_i that any node inside G_i is not isolated. We use this property in a proof of the achievable maximum broadcast rate of a lattice graph in section 4.5.3.1.

additional assumption of network model

For simplicity in proofs, the neighbor set R must include the node itself ($(0,0) \in R$). The following requirement should also be met:

Requirement 1 $\{(-1,0),(1,0),(0,-1),(0,1)\} \subset R$

An important assumption is that the *wireless broadcast advantage* is used: each transmission is overheard by several nodes. As a result, the graph is in reality a *(unit disk) hypergraph*: (it is slightly different from *random geometric graphs* [87] where links are independent).

4.5.2.2 Extended Rate Selection : IREN/IRON

The principle IREN/IRON sets the following transmission rates:

- IREN (Increased Rate for Exceptional Nodes): the rate of transmission is set to M, for the source node and all the border nodes (the "exceptional" nodes).
- IRON (Identical Rate for Other Nodes): every other node, except the source and all the border nodes, transmits with rate 1.

Figure 4.16(b) shows the selected rates using heuristic IREN/IRON for sample graph. These rates can be globally scaled by the same amount such as setting $k \times M$ for exceptional nodes and $k \times 1$ for other nodes where constant k is $k > 0$. When scaling these rates, both the transience cost and the achieved broadcast rate linearly increase. The linear increases keeps their ratio identical and hence global scaling of these rates does not affect energy efficiency.

The Rationale of IREN/IRON follows our intuitive reasoning of IRON except for nodes around the border. The intuitive reasoning of IRON assigns the identical rate 1 to most other nodes because most nodes have the same expected number of neighbors M. Then, most other nodes have an average receiving rate M that is the sum of transmission from their neighbors. For the source, its transmission rates should make the transmission rate of its neighbors 1 and the expected number of neighbors is M. Hence the source must transmit at least at the rate M, otherwise the transmission rate of the source would be a bottleneck. With this rate selection (IRON), only border nodes have lower receiving rates than M because they have smaller neighborhoods (less than M neighbors). Hence, by setting 1 transmission rate to the border nodes and their neighbors, the border nodes are guaranteed to receive a sufficient rate M. As networks become larger and denser, not only do most nodes have the expected neighbor number, but also the border nodes represent a minority of nodes. Hence, the transmission cost from the border nodes have limited impact on the efficiency. After tracking the above intuitive rationale of IREN/IRON rate selection, we raise a question whether the above rationale is sufficient to achieving broadcast rate of M at every node. As an answer to the question, we prove in section 4.5.2.3 that the maximum broadcast rate from IREN/IRON is always M at every node for a lattice graph and that the rate is asymptotically M for a random unit disk graph

4.5.2.3 Performance of IREN/IRON

Achievable Broadcast Rate: Min-Cut The essence of our main result is the following property proved in section 4.5.3.1, Theorem 5:

Property 1 *With the rate selection IREN/IRON, the min-cut of a lattice graph is exactly equal to $C_{\min} = M$ (with $M = |R| - 1$).*

For random unit disk graphs, by mapping the points to an imaginary lattice graph (*embedded lattice*) as an intermediary step, we are able to find bounds of the capacity of random unit disk graphs. This capacity bound is similar with the spirit of [84]. In [84], Salah et al modelled a wireless network using quasi random geometric graph and showed that the capacity of min-cut of a single source multicast with network coding is concentrated around the value np' where n is the number of relay nodes between a source and destination nodes and p' is the average connection probability. The value np' is approximately the expected number of neighbors, M as described in the following two properties.

Property 2 *The min-cut $C_{\min}(s)$ of the source s in a random unit disk graph $\mathcal{V}$, is bounded by the min-cut $C^{(\mathcal{L})}_{\min}(s_{\mathcal{L}})$ of some points in the embedded lattice $s_{\mathcal{L}}$ as follows: $C_{\min}(s) \geq m_{\min} C^{(\mathcal{L})}_{\min}(s_{\mathcal{L}})$ where $m_{\min}$ is a random variable related to the node number of a random unit disk graph mapped to one point of the embedded lattice.*

The above property described at section 4.5.3.2, Theorem 6 in detail, gives us a general implication that $m_{\min} C^{(\mathcal{L})}_{\min}(s_{\mathcal{L}})$ is generally "close" to M. Using the implication we prove the following property which is a deduced result for a random unit disk graph at section 4.5.3.2, Theorem 8.

Property 3 *Assume a fixed range. For a sequence of random unit disk graphs $(\mathcal{V}_i)$, with sources s_i, with size $L \to \infty$ and with a density $M \to \infty$ such as $M = \Omega(L^{\theta})$, for any fixed $\theta > 0$, we have the following convergence in probability: $\frac{C_{\min}(s)}{M} \xrightarrow{p} 1$.*

4.5.2.4 Performance: Transmission Cost per Broadcast

As described in section 4.3.2, the metric for cost is the number of (packet) transmissions per a (packet) broadcast from the source to the entire network. The transmission number per broadcast is denoted as E_{cost}. The E_{cost} with IREN/IRON rate selection can be deduced from the ratio of the transmission number that occurred inside a network per unit time to broadcasted packet number per the unit time. One parameter of E_{cost}, the broadcast packet number per unit time is computed from the achievable maximum broadcast rate, the min-cut $C_{\min}(s)$. The other parameter of E_{cost} is computed from the transmission rate of all nodes in the border area $A_{\Delta G}$ and interior area A_{G_i}. Namely E_{cost} is deduced from the min-cut $C_{\min}$, the border and interior the areas $A_{\Delta G}, A_{G_i}$, the associated node rates (along with the rates of the nodes on the border and in the interior) and the node density μ. For fixed W, $M, L \to \infty$:

$$\begin{aligned}
E_{\text{cost}} &= \frac{\sum_{v \in \mathcal{V}} C_v}{C_{\min}} = \frac{M|\delta G| + |G_i|}{C_{\min}} \\
&= \frac{\mu\left(M(L-W)4W + (L-2W)^2\right)}{C_{\min}} \\
&= \frac{1}{C_{\min}} \mu L^2 \left(1 + O(\frac{1}{L}) + \frac{4MW}{L}(1 + O(\frac{1}{L}))\right)
\end{aligned} \tag{4.6}$$

For random unit disk graphs $\mathcal{V}$, E_{cost} is an expected value $E_{\text{cost}} = E(E_{\text{cost}}(\mathcal{V}))$ with $\mu = \frac{N}{L^2}$, and for a lattice with $\mu = 1$.

4.5.2.5 Near Optimal Performance for a large Network

Sections 4.5.2.3 and 4.5.2.4 gave the performance and cost with the IREN/IRON principle. As indicated previously, for a given (hyper)graph, the optimal rate selection, and the optimal (minimum-cost) total rate of the network may be computed with a linear program [13]. The optimal cost is not easily computed, and in this section an indirect route is chosen, by using a bound.

Assume that every node has at most $M_{\max}$ neighbors: one single transmission can provide information to $M_{\max}$ nodes at most. Hence, in order to broadcast one original packet from the source to all N nodes, at least $E_{\text{bound}} = \frac{N}{M_{\max}}$ transmissions are necessary. The E_{bound} offers the standard to measure the energy efficient optimality. We compare the E_{bound} to E_{cost}, and represent the comparison results as the relative cost : $E_{\text{rel-cost}} =$

$\frac{E_{\text{cost}}}{E_{\text{bound}}} \geq 1$. We prove that $E_{\text{rel}-\text{cost}} \to 1$ for the following two networks : a lattice graph and a random unit disk graph

a Lattice Graph

For a lattice graph, we assume a constant range, and hence we have a constant neighborhood definition set R and a constant M which is the number of neighbors for any node not in the border area. The width of the border area W is lower than 2, and thus the border area includes all nodes whose distance from the edge of the network is less than 2. Because the size of the neighborhood is kept constant, the width of the border also stays constant. Therefore, for lattice graphs, the border width W, the neighborhood R and the expected number of neighbors M ($M = |R| - 1$) are fixed, and only the network size L increases to infinity. The increasing network size L only affects the total number of nodes N: $N = L^2$, and does not affect the network density μ and maximum number of neighbors $M_{\max}$: $\mu = 1$ and $M_{\max} = M$. Therefore from section 4.5.2.4 and from Property 1 E_{cost} for a lattice graph is :

$$\begin{aligned} E_{\text{rel}-\text{cost}} &= E_{\text{cost}} \frac{M_{\max}}{N} = \frac{\mu\left(M(L-W)4W + (L-2W)^2\right)}{C_{\min}} \frac{M_{\max}}{N} \\ &= \left((1 + O(\frac{1}{L}) + \frac{4MW}{L}(1 + O(\frac{1}{L}))\right) = 1 + O(\frac{1}{L}). \end{aligned} \tag{4.7}$$

Random Unit Disk Graphs

For random unit disk graphs, first notice that an increase of the density M does not improve the relative cost $E_{\text{rel}-\text{cost}}$ (due to the cost of border nodes).

Now consider a sequence of random graphs, as in Property 3, with fixed radio range ρ, fixed border width W, and size $L \to \infty$ and with a density $M \to \infty$ such as $M = \Omega(L^\theta$, for some arbitrary fixed $\theta > 0$, with the additional constraint that $\theta < 1$. We have:

$E_{\text{rel}-\text{cost}} = E_{\text{cost}} \frac{M_{max}}{N} = = \frac{M}{C_{\min}} \frac{M_{\max}}{M} \frac{\mu L^2}{N} \left(1 + O(\frac{1}{L}) + \frac{4MW}{L}(1 + O(\frac{1}{L}))\right)$.

Each of part of the product converges toward 1, in probability:

using Property 3, we have the convergence of $\frac{C_{\min}}{M} \xrightarrow{p} 1$, when $L \to \infty$ and similarly with Theorem 8 we have $\frac{M_{\max}}{M} \xrightarrow{p} 1$. By definition $N = \mu L^2$. Finally, $M = \Omega(L^\theta)$ for $\theta < 1$ implies that $\frac{4MW}{L} \to 0$.

As a result we have: $E_{\text{rel}-\text{cost}} \xrightarrow{p} 1$ in probability, when $L \to \infty$.

Random Unit Disk Graphs without Network Coding

In order to compare the results that are obtained when network coding is not used, one can reuse the argument of [73] as explained in section 4.3.2.2. Consider the broadcasting of one packet. Consider one node of the network that has repeated the packets. It must have received the transmission from another connected neighbor. In a unit disk graph, these two connected neighbors share a neighborhood area at least equal to $(\frac{2\pi}{3} - \frac{\sqrt{3}}{2})\rho^2$, and every node lying within that area will receive duplicates of the packets. Considering this inefficiency, for dense unit disk graphs one can deduce the following bound: $E^{(\text{no-coding})}_{\text{rel-cost}} \geq \frac{6\pi}{2\pi + 3\sqrt{3}}$ Notice that $\frac{6\pi}{2\pi+3\sqrt{3}} \approx 1.6420\ldots > 1$.

Near Optimality The asymptotic optimality is a consequence of the convergence of the cost bound $E_{\text{rel-cost}}$ toward 1. Since it is not possible to have a relative cost $E_{\text{rel-cost}}$ lower than 1, the rate selection IREN/IRON is asymptotically optimal for the two cases presented when $L \to \infty$. Note that, this indirect proof is in fact a stronger statement than optimality of the rate selection in terms of energy-efficiency: it exhibits the fact that asymptotically (nearly) all the transmissions will be *innovative* for the receivers. Note that it is not the case in general for a given instance of a hypergraph. It evidences the following remarkable fact for the large homogeneous networks considered: network coding may be achieving not only optimal efficiency, but also, asymptotically, perfect efficiency — achieving the information-theoretic bound for each transmission.

Notice that traditional broadcast methods without network coding (such as the ones based on connected dominating sets) cannot achieve this efficiency, since their lower bound is 1.642.

4.5.3 Proofs of the Achievable Capacity with Network Coding in 2D networks

In this section, we provide a formal proof for both Property 1 and Property 3 of section 4.5.2.3.

4.5.3.1 Proof for a Lattice Graph

Overview of the Proof We first start with a proof for a lattice graph (such as the one Figure 4.1(b)). Our objective is to prove Theorem 5 (section 4.5.3.1), which indicates that for one source s, the min-cut $C_{\min}$ of the lattice graph is M (with IREN/IRON).

In order to compute the global min-cut $C_{\min}(s)$, we start with considering one destination node t in the network, and we provide a bound the min-cut of the (hyper)-graph between s and t, that is, $C_{\min}(s, t)$.

The proof proceeds as follows: we first link the capacity of the cut between nodes in S and nodes in T with the number of nodes in S which are neighbors of nodes in T. The number of these nodes decides the capacity of the cut. To compute the number of these nodes, we use the fact that the neighborhood is obtained with a Minkowski sum. Precisely, the number of neighbors is computed by applying the inequality on Minkowski sums. As a consequence, we computes the capacity of the cut by computing the number of neighbors with the inequality on Minkowski sums. However, because of the effect from the border $\Delta\mathcal{L}$, several special cases exist for applying the inequality. We classify each case into three categories, and for each category we prove that the capacity of the cut has the desired bound, the expected number of neighbors M. In the proof, we explain the discrete geometry property of neighborhood which is defined in section 4.3.1 and section 4.5.2.1.

Preliminaries Let Γ be a full (with full rank) , unbounded, *integer (point) lattice* in n-dimensional space; Γ is the set $\mathbb{Z}^n$, where the lattice points are n-tuples of integers.

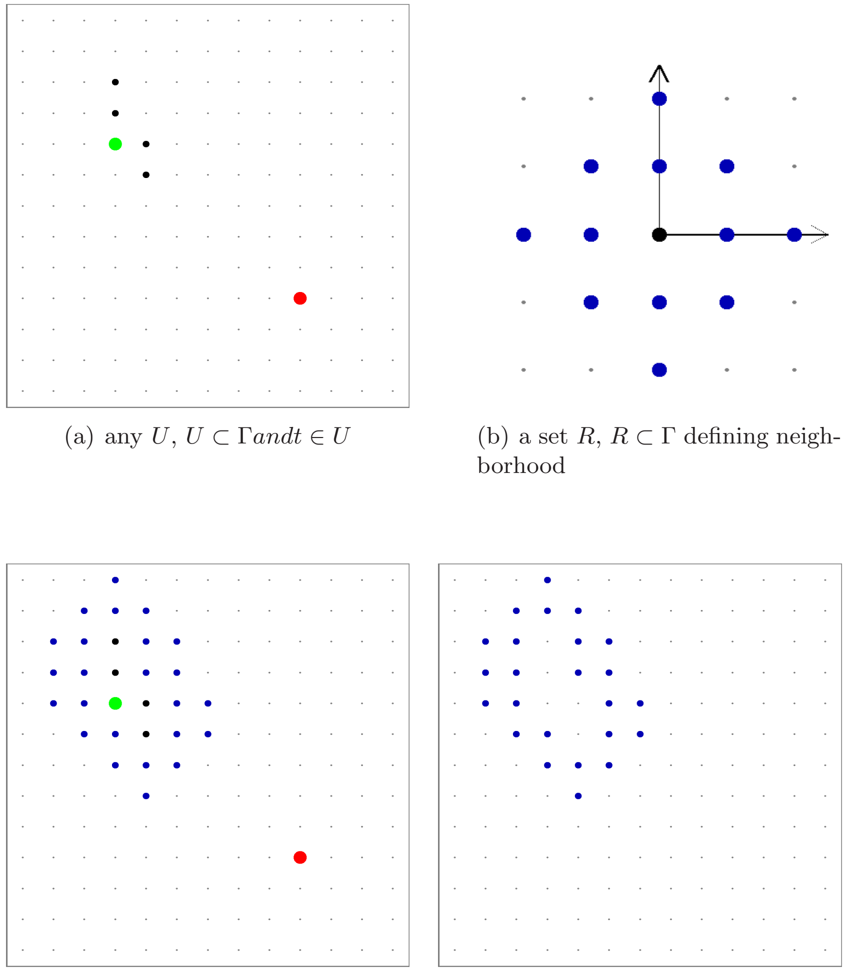
For a lattice graph, only points on the full lattice are relevant; therefore, in this section, the notations $\mathcal{L}, \mathcal{L}_i, \Delta\mathcal{L}$ explained in section 4.5.2.1 are used, for the parts of the full lattice Γ that are in $G, G_i, \Delta G$ respectively.

The proof is based on the use of the Minkowski addition, and a specific property of discrete geometry (4.8) below. The Minkowski addition is a classical way to express the neighborhood of one area (for instance, see [88] and Figure 3(a), and Figure 4 of that reference).

Given two sets A and B of $\mathbb{R}^n$, the Minkowski sum of the two sets $A \oplus B$ is defined as the set of all vector sums generated by all pairs of points in A and B, respectively:
$A \oplus B \triangleq \{a + b : a \in A, b \in B\}$

Consider a subset R of Γ, defining neighborhood, such as the ones in Figure 4.2(a) and Figure 4.2(b), with an origin at the point $(0, 0)$. We denote this set R as the *lattice neighborhood definition set*. Then the set of neighbors $\mathcal{N}(t)$ of one node t, with t itself, is:
$\mathcal{N}(t) \cup \{t\} = \{t\} \oplus R$
The neighbors of t are given with:
$\mathcal{N}(t) = (\{t\} \oplus R) \setminus \{t\}$.

In a similar way, any subset U, $U \subset \Gamma$ and $t \in U$, the neighbors of U are given with $(U \oplus R) \setminus U$ as seen in Figure 4.17.



(a) any U, $U \subset \Gamma and t \in U$

(b) a set R, $R \subset \Gamma$ defining neighborhood

(c) Minkowski sum of U and R, $U \oplus R$

(d) neighborhood of U, $(U \oplus R) \setminus U$

Figure 4.17: Neighborhood obtained using the Minkowski sum

Rewriting the neighborhood in terms of the Minkowski sum as seen in Figure 4.17 enables us to use *Brunn-Minkowski-Lysternik inequalities* which gives a bound on the size of the Minkowski sum of two compact sets of $\mathbb{R}^n$; for integer lattice, several integer variants exist including the following one [89]: for two subsets A, B of the integer lattice $\mathbb{Z}^n$,

$$|A \oplus B| \geq |A| + |B| - 1 \tag{4.8}$$

where $|X|$ represents the number of elements of a subset X of $\mathbb{Z}^n$

Bound on the capacity of one cut $C(S)$

Consider a lattice $\mathcal{L}$ and a source s. We start with the definition of $C_{\min}(s,t)$ of (3.2): it requires considering the capacities of every s-t-cut S,T. Let $C(S)$ be the capacity of such a s-t cut $S,T \in Q(s,t)$.

We have the following lemma linking the capacity of the cut and the size of ΔS, the set of nodes of S which are neighbors of nodes of T

Lemma 2 $C(S) \geq |\Delta S|$ *(with* $\Delta S \triangleq \{v \in S : \mathcal{N}(v) \cap T \neq \emptyset\}$ *defined in (3.4))*

Proof: $C_v \geq 1$ with IREN/IRON and with (3.4), $C(S) = \sum_{v \in \Delta S} C_v$

Lemma 3 *If* $U \subset \mathcal{L}_i$ *then* $U \oplus R \subset \mathcal{L}$

Proof: The requirement on W in section 4.5.2.1 translates into: for any node $x \in \mathcal{L}_i$, $\{x\} \oplus R \subset \mathcal{L}$, hence the result.

Lemma 4 *When the following requirement for* R *is satisfied, for any two nodes* u, v *inside the border area, a path exists only connecting points for the border area.*
The requirement for R*, (Requirement 4) is :*

> *The set* R *is a subset of* Γ *and should include the origin point* $(0,0)$ *as well as the 4 points which are immediate neighbors on the lattice:* $(1,0)$, $(0,1)$, $(-1,0)$, $(0,-1)$

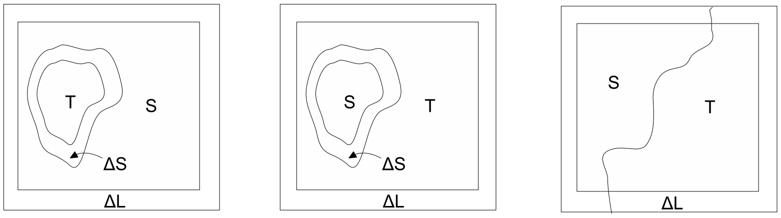
Proof: Because of the requirement for the set R, R should include the 4 immediate neighbors in the directions "left, right, up, and down". This neighborhood prevents isolated node inside the border area and makes the border area as a connected area. Hence the lemma follows.

Theorem 4 *The capacity of one cut* $C(S)$ *is such that:* $C(S) \geq M$

Proof: There are three possible cases, either the set T has no common nodes with the border $\Delta\mathcal{L}$, or T includes all nodes of $\Delta\mathcal{L}$, or finally T includes only some of the nodes in the border area. Formally:

- First case: $T \cap \Delta\mathcal{L} = \emptyset$
- Second case: $\Delta\mathcal{L} \subset T$
- Third case: $T \cap \Delta\mathcal{L} \neq \emptyset$ and $\Delta\mathcal{L} \not\subset T$

We will prove inequality of theorem 4 in all three cases.



(a) $T \cap \Delta\mathcal{L} = \emptyset$ (b) $\Delta\mathcal{L} \subset T$ (c) $T \cap \Delta\mathcal{L} \neq \emptyset$ and $\Delta\mathcal{L} \not\subset T$

Figure 4.18: S/T partition of a lattice network with $\Delta\mathcal{L}$

First case, $T \cap \Delta\mathcal{L} = \emptyset$:

With Lemma 3, we know that $T \oplus R \subset \mathcal{L}$, hence we can effectively write the neighbors of nodes in T as the Minkowski addition (without getting points in Γ but out of $\mathcal{L}$): $\Delta T \triangleq (T \oplus R) \setminus T$

It follows that: $|\Delta T| \geq |T \oplus R| - |T|$ Now the inequality (4.8) can be used: $|T \oplus R| \geq |T| + |R| - 1$

Hence we get: $|\Delta T| \geq |T| + |R| - 1 - |T|$, and therefore:

$$|\Delta T| \geq |R| - 1 \tag{4.9}$$

Recall that S and T form a partition of $\mathcal{L}$; and since ΔT is a subset of $\mathcal{L}$, by definition without any point of T, we have $\Delta T \subset S$. Hence actually $\Delta T \subset \Delta S$ (with the definition of ΔS in (3.4)). We can combine this fact with Lemma 2 , Lemma 3,and (4.9), to get:

$|C(S)| \geq |R| - 1$ and Theorem 4 is proved for the first case.

Second case, $\Delta\mathcal{L} \subset T$:

In this case all the points of the border area are included in T. As a consequence, the complementary set of T, S has no nodes on the border, i.e. $S \cap \Delta\mathcal{L} = \emptyset$ and it results in $S \subset \mathcal{L}_i$.

We show that a set S has an equal or greater number of nodes that are neighbors of nodes in T than $|R| - 1$. The method to prove it is similar to the method to prove the first case, but we consider neighborhood in the opposite way: we consider the nodes in S that are neighbors of nodes in T.

Let us denote S_i the "interior" of S, that is, the set of nodes of S whose points are not within range of the set T Precisely:

$S_i \triangleq \{x : x \in S \text{ and } (x \oplus R) \cap T = \emptyset\}$

By definition of ΔS in eq. (3.4), ΔS is the sets of nodes of S which are within range of the set T, and hence the subsets S_i and ΔS form a partition of S

Additionally, because $S_i \subset S$ and $S \subset \mathcal{L}_i$, we know with lemma 3 that $S_i \oplus R \subset \mathcal{L}$. Since by definition of S, $S_i \oplus R$ has no common element with T, and since S and T are a partition of $\mathcal{L}$, the property follows:[1]

$$S_i \oplus R \subset S \tag{4.10}$$

Now there are two possibilities: either $S_i = \emptyset$ or not.

• If $S_i = \emptyset$, the implication is that $S = \Delta S$, hence in particular, $s \in \Delta S$. Going back to the definition of a cut in (3.4), we had:

$C(S) = \sum_{v \in \Delta S} C_v$ by definition,

$\Rightarrow C(S) \geq C_s$ because $s \in \Delta S$

$\Rightarrow C(S) \geq M$ because $C_s = M$ with IREN/IRON. and the theorem 4 is proved when $\Delta\mathcal{L} \subset T$ and $S_i = \emptyset$.

• Otherwise, $S_i \neq \emptyset$. Starting from eq. 4.10, we had:

$S_i \oplus R \subset S$

$\Rightarrow |S| \geq |S_i \oplus R|$, and as a result, with ineq. 4.8:

$$|S| \geq |S_i| + |R| - 1 \tag{4.11}$$

Because S_i and ΔS form S, $\Delta S = S \setminus S_i$

$\Rightarrow |\Delta S| \geq |S| - |S_i|$

$\Rightarrow |\Delta S| \geq |R| - 1$

Therefore with lemma 2, we deduce the capacity of the cut is such that:

$C(S) \geq |R| - 1$

and the theorem 4 is proved when $\Delta\mathcal{L} \subset T$ and $S_i \neq \emptyset$.

[1] Alternatively the reader familiar with mathematical morphology [90] could notice that S_i is the *erosion* of S by the structural element R. As a result $S_i \oplus R$ is actually the *opening* of S, and the following property of the opening is known: $S_i \oplus R \subset S$ (see [91] p.40).

Third case: $T \cap \Delta\mathcal{L} \neq \emptyset$ and $\Delta\mathcal{L} \not\subset T$:

Because T and S are a partition of $\mathcal{L}$, we deduce that $S \cap \Delta\mathcal{L} \neq \emptyset$; hence both T and S have nodes in the border area $\Delta\mathcal{L}$.

Let us consider such nodes: $u_t \in T \cap \Delta\mathcal{L}$ and $u_s \in S \cap \Delta\mathcal{L}$. With Lemma 3, there exist a path from u_s to u_t only connecting nodes in the border area.

Let us start with u_s, and iterate on the nodes of the path. Since u_s is in S and u_t is in T, we ultimately find a node u on the path from u_s to u_t such that u is still in S and that its successor v on the path is not in S but in T). By definition of ΔS, $u \in \Delta S$, and also $u \in \Delta\mathcal{L}$ by property of the path.

Hence now, the contribution of u to the capacity of the cut $C(S)$ can be used: $C(S) = \sum_{v \in \Delta S} C_v$ (from def. 3.4)
$\Rightarrow C(S) \geq C_u$, because $u \in \Delta S$
$\Rightarrow C(S) \geq M$ because $C_u = M$

and the theorem 4 is proved for the third case. □

Value of the Min-cut $C_{\min}(s)$ The results of the previous section immediately result in a property on the capacity of every s-t min-cut:

Theorem 5 *For any $t \in \mathcal{L}$ different from the source s:*
$C_{\min}(s,t) = M$ *; and as a result:* $C_{\min}(s) = M$

Proof: From theorem 5 we have the capacity of every s-t cut S/T verifying $C(S) \geq M$ and hence $C_{\min}(s,t) \geq M$.

Conversely let us consider a specific cut, $S_s = \{s\}$ and $T_s = \mathcal{L} \setminus \{s\}$. Obviously s has at least one neighbor, which has to be in T, hence $\Delta S = \{s\}$. The capacity of the cut is $C(S_s) = \sum_{v \in \Delta S} C_v = C_s = M$ and thus $C_{\min}(s,t) \leq M$. As a result, the theorem follows. □

4.5.3.2 Proof of the Value of Min-Cut for Unit Disk Graphs

In this section, we prove a probabilistic result on the min-cut of random unit disk graphs using an virtual "embedded" lattice. The random unit graph is denoted as $\mathcal{V}$, whereas the embedded lattice is denoted as $\mathcal{L}$ along with $\Delta\mathcal{L}$ and $\mathcal{L}_i$ using the notations in section 4.5.3;.

The elements of $\mathcal{V}$ are still called "nodes", but the elements of $\mathcal{L}$ are called "points" to emphasize that they are virtual. We assume $W > \rho$ (for instance $W = 2\rho$)

Embedded Lattice Given the $L \times L$ square area G, we start from fitting a *rescaled* lattice inside G, with a scaling factor r. Precisely, the rescaled lattice is fitted so that it intersects the square area G and the set $\{(rx, ry) : (x, y) \in \mathbb{Z}^2\}$. After intersecting them, we map the points of G (black circles in Figure 4.19) to the closest point (blue circles in Figure 4.19) on the rescaled lattice $\mathcal{L}$. The mapping is done through $\lambda(x)$ which transforms a point u of the Euclidian space $\mathbb{R}^2$ to the closest point of $\mathcal{L}$. Formally, for $u = (x, y) \in \mathbb{Z}^2$, $\lambda(x) \triangleq (r\lfloor \frac{x}{r} + \frac{1}{2} \rfloor, r\lfloor \frac{y}{r} + \frac{1}{2} \rfloor)$.

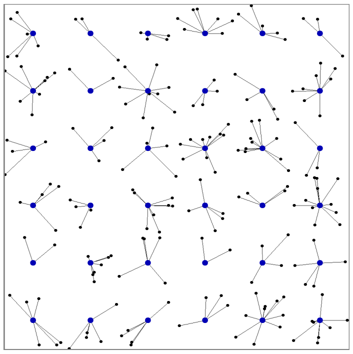


Figure 4.19: Mapping points in G to a point in embedded lattice $\mathcal{L}$

For $u \in \mathcal{L}$, $\lambda^{-1}(u)$ is the set of nodes of $\mathcal{V}$ that are mapped to u. When transforming, the area of $\mathbb{R}^2$ that is mapped to a same point of the lattice, is a $r \times r$ square around that point. We choose r so that G fits exactly so that any $r \times r$ square is not truncated. The exact fitting is achieved by taking the origin point of $\mathbb{R}^2$ as the center of the square G, and by selecting $r = \frac{2k+1}{L}$ where k is a positive integer.

Let u be a point of the lattice $\mathcal{L}$, and let denote $m(u)$ the number of points of $\mathcal{V}$ on G that are mapped to u on $\mathcal{L}$. The points on G are in the square around u and $m(u) \triangleq |\lambda^{-1}(u)|)$. Since G is a random unit disk graph and $\mathcal{V}$ is nodes in G, $m(u)$ is a random variable and the minimum value and maximum value of $m(u)$ is denoted as :

$m_{\min} \triangleq \min_{u \in \mathcal{L}} m(u)$ and $m_{\max} \triangleq \max_{u \in \mathcal{L}} m(u)$

Neighborhood of the Embedded Lattice In order to find the capacity of the cut of a random unit disk graph using an embedded lattice, we first define the neighborhood R for the embedded lattice. The desired property is to find the relationship between neighborhoods on the unit graph and neighborhoods on the embedded lattice that is obtained by mapping nodes' position on the unit graph to points.

In order to find the relationship, we choose the points of the lattice inside a disk of radius $\rho - 2r$ as the neighborhood of the embedded lattice $R(r)$ as : $R(r) = \{(rx, ry) : (rx)^2 + (ry)^2 \leq (\rho - 2r)^2; (x, y) \in \mathbb{Z}^2\}$

With the selected neighborhood $R(r)$, we have the following desired property.

Lemma 5 *Let us consider two nodes of u, v of $\mathcal{V}$ that are mapped on the lattice $\mathcal{L}$ to $u_{\mathcal{L}}$ and $v_{\mathcal{L}}$ respectively:*

- *if $u_{\mathcal{L}}$ and $v_{\mathcal{L}}$ are neighbors on the lattice, then u and v are neighbors on the graph $\mathcal{V}$*

Proof: Using triangle inequality of the Euclidian distance $\parallel . \parallel$ we have two inequations : $\parallel u - u_{\mathcal{L}} \parallel + \parallel u_{\mathcal{L}} - v_{\mathcal{L}} \parallel \geq \parallel u - v_{\mathcal{L}} \parallel$ and $\parallel u - v_{\mathcal{L}} \parallel + \parallel v - v_{\mathcal{L}} \parallel \geq \parallel u - v \parallel$. From the two inequations we have $\parallel u - v \parallel \leq \parallel u - u_{\mathcal{L}} \parallel + \parallel u_{\mathcal{L}} - v_{\mathcal{L}} \parallel + \parallel v_{\mathcal{L}} - v \parallel$.

By definition of neighborhood on the lattice, $u_{\mathcal{L}} - v_{\mathcal{L}} \in R(r)$, hence, $\parallel u_{\mathcal{L}} - v_{\mathcal{L}} \parallel \leq \rho - 2r$

Moreover since $u_{\mathcal{L}}$ is the closest point on the lattice of u, and we have $\parallel u - u_{\mathcal{L}} \parallel \leq \frac{\sqrt{2}}{2} r$ (the length of the half-diagonal of an $r \times r$ square), which is implies $\parallel u - u_{\mathcal{L}} \parallel \leq r$. The same reasoning applies to v and $v_L at$. As a result we have :
$\parallel u - v \parallel \leq 2r + \parallel u_{\mathcal{L}} - v_{\mathcal{L}} \parallel \leq \rho$. Hence the lemma follows. □

Lemma 6 $|R(r)| \leq \pi \frac{\rho^2}{r^2}$

Proof: In a spirit similar to the mapping to the lattice, let us consider the square of size $r \times r$ around each point of $R(r)$. Such squares are disjoint for different points of $R(r)$; let us denote $\hat{R(r)}$ the union of all such squares of every point of $R(r)$.

We have, for every point of $u \in \hat{R(r)}$: there exists a point $v \in R(r)$ such that u is in the square around v. Then by a similar argument to lemma 5, $\parallel u - v \parallel \leq \frac{\sqrt{2}}{2} r \leq r$; in addition $\parallel v \parallel \leq \rho - 2r$, from the definition of $R(r)$. Therefore $\parallel u \parallel \leq \rho$, hence $\hat{R(r)}$ is included in the disk of radius ρ. Therefore its area $A(\hat{R(r)})$ verifies $A(\hat{R(r)}) \leq \pi \rho^2$.

In addition, by definition of $\hat{R(r)}$ as union of disjoint squares, we also have another expression of its area: $A(\hat{R(r)}) = |R(r)| r^2$. Using this equality with the previous inequality of $A(\hat{R(r)})$ gives the result. □

Lemma 7 $|R(r)| = \pi \frac{\rho^2}{r^2} + O(\frac{1}{r})$ *when* $r \to 0$,

Proof: We can rewrite the definition of $R(r)$ as:

$$R(r) = \{(rx, ry) : x^2 + y^2 \leq (\frac{\rho - 2r}{r})^2; (x, y) \in \mathbb{Z}^2\} \tag{4.12}$$

The number of points in $|R(r)|$ is the number of lattice points within a circle with a radius fixed around the origin (the "*circle problem*"). From [91] p. 133, Gauß has shown that $N_c(d) = \pi d^2 + O(d)$, for a circle of radius d, when $d \to \infty$. Here $d = \frac{\rho}{r} - 2$, hence $|R(r)| = \pi(\frac{\rho}{r})^2 + O(\frac{1}{r})$, and the lemma follows. □

Relationship between the Capacities of the Cuts of the Embedded Lattice and the Random Disk Unit Graph The idea here is to show the relationship with a cut of the random unit graph, and a cut of the lattice graph.

Let us consider one source $s \in \mathcal{V}$, one destination $t \in \mathcal{V}$ and the capacity of any S/T cut. Every node of S and T is then mapped to the nearest point of the embedded lattice. For the source, we denote: $s_{\mathcal{L}} = \lambda(s)$.

An *induced cut* of the embedded lattice is constructed as follows:

- The border area width $W_{\mathcal{L}}$ is selected so as to be the greatest integer multiple of r which is smaller than W ; and $r < W - \rho$, so that requirement 1 of section 4.5.2.1 is met.

- For any point of the lattice $v_{\mathcal{L}} \in \mathcal{L}$, the rate $C^{(\mathcal{L})}_{v_{\mathcal{L}}}$ is set according to IREN/IRON on the lattice: $C^{(\mathcal{L})}_{v_{\mathcal{L}}} = |R(r)| - 1$ when $v_{\mathcal{L}}$ is within the border area of width $W_{\mathcal{L}}$, and $C^{(\mathcal{L})}_{v_{\mathcal{L}}} = 1$ otherwise.

- $S_{\mathcal{L}}$ is the set including the point $s_{\mathcal{L}}$, and the points of the lattice $\mathcal{L}$ such as only nodes of S are mapped to them:

$$S_{\mathcal{L}} \triangleq \{s_{\mathcal{L}}\} \cup \{u_{\mathcal{L}} : \lambda^{-1}(u_{\mathcal{L}}) \subset S \tag{4.13}$$

- $T_{\mathcal{L}}$ is the set of the rest of points of $\mathcal{L}$.

Note that $t \in T_{\mathcal{L}}$; that all the points of the lattice, to which both points from S and T are mapped, are in $T_{\mathcal{L}}$; and that the points to which no points are mapped are in $S_{\mathcal{L}}$: $S_{\mathcal{L}}/T_{\mathcal{L}}$ is indeed a partition and a $s_{\mathcal{L}} - t_{\mathcal{L}}$ cut.

Recall the definition of a cut in Equation 3.4, we have:
$C(S) = \sum_{v \in \Delta S} C_v$ and $C^{(\mathcal{L})}(S_{\mathcal{L}}) = \sum_{v \in \Delta S_{\mathcal{L}}} C_v$ where ΔS and $\Delta S_{\mathcal{L}}$ are subsets of S and $S_{\mathcal{L}}$ respectively.

We have the following relationship between these two sets:

Lemma 8 *Excluding $s_{\mathcal{L}}$ and s, the nodes of $\mathcal{V}$ that are mapped to points of $\Delta S_{\mathcal{L}}$, are in ΔS ; that is:*

$$\lambda^{-1}(\Delta S_{\mathcal{L}} \setminus \{s_{\mathcal{L}}\}) \subset \Delta S \setminus \{s\}$$

Proof: $\lambda^{-1}(\Delta S_{\mathcal{L}}) = \cup_{u_{\mathcal{L}} \in \Delta S_{\mathcal{L}}} \lambda^{-1}(u_{\mathcal{L}})$ hence it is sufficient to prove the property for $\lambda^{-1}(u_{\mathcal{L}})$ for every $u_{\mathcal{L}} \in \Delta S_{\mathcal{L}}$.

Let us consider one such point $u_{\mathcal{L}} \in \Delta S_{\mathcal{L}} \setminus \{s_{\mathcal{L}}\}$. By definition of $\Delta S_{\mathcal{L}}$, there exists a point $v_{\mathcal{L}} \in T_{\mathcal{L}}$ within range for $\mathcal{L}$ (that is: $(u_{\mathcal{L}} - v_{\mathcal{L}}) \in R(r)$).

If $\lambda^{-1}(u_{\mathcal{L}}) = \emptyset$, then the property $\lambda^{-1}(u_{\mathcal{L}}) \in \Delta S \setminus \{s\}$ is verified. Hence let us consider the case where $\lambda^{-1}(u_{\mathcal{L}}) \neq \emptyset$:

Since $v_{\mathcal{L}} \in T_{\mathcal{L}}$, by the definition of this set, there exists at least one node of T mapped to $v_{\mathcal{L}}$ and thus: $\lambda^{-1}(v_{\mathcal{L}}) \cap T \neq \emptyset$.

Now consider two points of these non-empty sets, $u \in \lambda^{-1}(u_{\mathcal{L}})$ and $v \in \lambda^{-1}(v_{\mathcal{L}}) \cap T$:

- From Lemma 5, we know that u and v are within range ($\| u - v \| \leq \rho$).
- Recall that $\Delta S_{\mathcal{L}} \subset S_{\mathcal{L}}$. By definition of $S_{\mathcal{L}}$, since $u_{\mathcal{L}}$ is in $S_{\mathcal{L}}$, u must be in S.
- $v \in T$

These three conditions imply that $u \in \Delta S$. Also s is mapped to the unique $\lambda(s) = s_{\mathcal{L}}$; therefore $u_{\mathcal{L}} \neq s_{\mathcal{L}}$ implies $u \neq \Delta S \setminus \{s\}$. It follows that $\lambda^{-1}(u_{\mathcal{L}}) \subset \Delta S \setminus \{s\}$, and, as a consequence, the lemma 8

It is now possible to use this subset of ΔS to prove the following lemma on relating the cut of $\mathcal{V}$ and its induced cut:

Lemma 9 *The capacity $C(S)$ of the cut S/T and the capacity of the induced cut $C^{(\mathcal{L})}(S_{\mathcal{L}})$ verify: $C(S) \geq m_{\min} C^{(\mathcal{L})}(S_{\mathcal{L}})$*

Proof: First note that if $m_{\min} = 0$, the lemma is proved. Hence, in the rest of the proof, we can assume that this integer verifies $m_{\min} \geq 1$.

In this case, notice that there are $\frac{L^2}{r^2}$ squares of size $r \times r$, each with at least $m_{\min}$ nodes, therefore the total number of nodes verifies: $N \geq \frac{L^2}{r^2} m_{\min}$, and then $\mu \geq \frac{1}{r^2} m_{\min}$ by definition of μ. Combining this with lemma 6, we get:

$$|R(r)| \leq \pi\rho^2 \mu m_{\text{min}} \tag{4.14}$$

Now consider again the definition of a cut in Equation 3.4. There are two cases: one without the source, with the source:

$C(S) = \sum_{v \in \Delta S} C_v = \sum_{v \in \Delta S \setminus \{s\}} C_v + \sum_{v \in \Delta S \cap \{s\}} C_v$

With Lemma 8, we know a subset of ΔS, hence:

$C(S) \geq C_{\text{interm.}} + C_{\text{src}}$

with $C_{\text{interm.}} = \sum_{v \in \lambda^{-1}(\Delta S_{\mathcal{L}} \setminus \{s_{\mathcal{L}}\})} C_v$ and $C_{\text{src}} = \sum_{v \in \Delta S \cap \{s\}} C_v$

• The first sum $C_{\text{interm.}}$ can be rewritten as:

$C_{\text{interm.}} = \sum_{v_{\mathcal{L}} \in \Delta S_{\mathcal{L}} \setminus \{s_{\mathcal{L}}\}} \sum_{v \in \lambda^{-1}(v_{\mathcal{L}})} C_v$

Let us consider all the nodes in the square area $\lambda^{-1}(v_{\mathcal{L}})$, and their rates compared to the rate of $v_{\mathcal{L}}$:

- If $C^{(\mathcal{L})}_{v_{\mathcal{L}}} = 1$, then since IREN/IRON assigns only rates $\geq$ 1, we have $C_v \geq C^{(\mathcal{L})}_{v_{\mathcal{L}}}$ for any $v \in \mathcal{V}$.

- If $C^{(\mathcal{L})}_{v_{\mathcal{L}}} > 1$, $v_{\mathcal{L}}$ is a border node for $\mathcal{L}$ (and $W_{\mathcal{L}}$), and $C^{(\mathcal{L})}_{v_{\mathcal{L}}}$ must actually be $|R(r)| - 1$. Since $W_{\mathcal{L}}$ is chosen so that $W_{\mathcal{L}} < W$, we have also: $\lambda^{-1}(v_{\mathcal{L}})$ is a set of border nodes of $\mathcal{V}$. Their rate is $C_v = \pi\rho^2 M$ by definition.

  From Equation 4.14, we have $C_v \geq m_{\text{min}}|R(r)|$, hence $C_v \geq |R(r)|$, and finally: $C_v \geq C^{(\mathcal{L})}_{v_{\mathcal{L}}}$

As a result, in both cases, $\forall v \in \lambda^{-1}(v_{\mathcal{L}})$, $C_v \geq C^{(\mathcal{L})}_{v_{\mathcal{L}}}$, and:

$\sum_{v \in \lambda^{-1}(v_{\mathcal{L}})} C_{v_{\mathcal{L}}} = |\lambda^{-1}(v_{\mathcal{L}})| C^{(\mathcal{L})}_{v_{\mathcal{L}}} \geq m_{\text{min}} C^{(\mathcal{L})}_{v_{\mathcal{L}}}$

Hence $C_{\text{interm.}}(S) \geq m_{\text{min}} \sum_{v_{\mathcal{L}} \in \Delta S_{\mathcal{L}} \setminus \{s_{\mathcal{L}}\}} C^{(\mathcal{L})}_{v_{\mathcal{L}}}$

• The second sum C_{src} reduces to 0 or 1 term:

- If $s_{\mathcal{L}} \in \Delta S_{\mathcal{L}}$, then $\Delta S_{\mathcal{L}} \cap \{s_{\mathcal{L}}\} = \{s_{\mathcal{L}}\}$.

  With the same reasoning as in the proof of Lemma 8, necessarily $s \in \Delta S$ as well, and: $\Delta S \cap \{s\} = \{s\}$.

  $C_{\text{src}} = C_s = \pi\rho^2\mu$. As before, from Equation 4.14, we get $C_s \geq m_{\text{min}}|R(r)|$, hence $C_{\text{src}} \geq m_{\text{min}} C^{(\mathcal{L})}_{s_{\mathcal{L}}}$

- If $s_{\mathcal{L}} \notin \Delta S_{\mathcal{L}}$, then $\Delta S_{\mathcal{L}} \cap \{s_{\mathcal{L}}\} = \emptyset$, and obviously $\sum_{v_{\mathcal{L}} \in \Delta S_{\mathcal{L}} \cap \{s_{\mathcal{L}}\}} C_{v_{\mathcal{L}}}^{(\mathcal{L})} = 0$

In both cases, $C_{\text{src}} \geq m_{\min} \sum_{v_{\mathcal{L}} \in \Delta S_{\mathcal{L}} \cap \{s_{\mathcal{L}}\}} C_{v_{\mathcal{L}}}^{(\mathcal{L})}$

Putting together both inequalities for $C_{\text{interm.}}$ and C_{src}, the result is:

$C(S) \geq C_{\text{interm.}} + C_{\text{src}} \geq m_{\min} \sum_{v_{\mathcal{L}} \in \Delta S_{\mathcal{L}} \setminus \{s_{\mathcal{L}}\}} C_{v_{\mathcal{L}}}^{(\mathcal{L})} + m_{\min} \sum_{v_{\mathcal{L}} \in \Delta S_{\mathcal{L}} \cap \{s_{\mathcal{L}}\}} C_{v_{\mathcal{L}}}^{(\mathcal{L})}$

Hence: $C(S) \geq m_{\min} \sum_{v_{\mathcal{L}} \in \Delta S_{\mathcal{L}}} C_{v_{\mathcal{L}}}^{(\mathcal{L})}$

The right part of the inequality is actually the definition of the capacity of the $s_{\mathcal{L}} - t_{\mathcal{L}}$-cut, hence: $C(S) \geq m_{\min} C^{(\mathcal{L})}(S_{\mathcal{L}})$, which is the lemma.

Theorem 6 *The min-cut $C_{\min}(s)$ of the graph $\mathcal{V}$, verifies:*

$C_{\min}(s) \geq m_{\min}(|R(r)| - 1)$

Proof: From Lemma 9, any cut $C(S)$ is lower bounded by $m_{\min} C^{(\mathcal{L})}(S_{\mathcal{L}})$. Since $C^{(\mathcal{L})}(S_{\mathcal{L}})$ is the capacity of a cut of a lattice with IREN/IRON, Theorem 5 also indicates that: $C^{(\mathcal{L})}(S_{\mathcal{L}}) \geq C_{\min}^{(\mathcal{L})}(s_{\mathcal{L}}) = |R(r)| - 1$. Hence the lower bound $m_{\min}(|R(r)| - 1)$ for any $C(S)$, and therefore for the min-cut $C_{\min}(s)$.

Nodes of $\mathcal{V}$ Mapped to One Lattice Point In Theorem 6, $m_{\min}$ plays a central part.

In this section, a probabilistic bound is given for the variation of $m_{\min}$.

Theorem 7 *When $L \to \infty$ and $\mu \to \infty$,*

$\frac{m_{\min}}{\mu r^2} \xrightarrow{p} 1,$

and $\frac{m_{\max}}{\mu r^2} \xrightarrow{p} 1.$

Proof: When a position of a node is i.i.d, for all i i $\in \{1, .., N\}$, $m(u_{\mathcal{L}})$ is $X = Y_1 + Y_2 + ...Y_{N-1}$, where Y_i is i.i.d Bernoulli trials of probability p, and $p = \frac{r^2}{L^2}$ is probability for one node to be mapped to $u_{\mathcal{L}}$. ($0 \leq p \leq 1$, r is the scaling factor of the neighborhood of the embedded lattice, L is the length of a square network).

Let X be the number of nodes to be mapped to $u_{\mathcal{L}}$. Because X is a random variable, $P(X < a) = P(e^{-\lambda X} > e^{-\lambda a}) < \frac{E(e^{-\lambda X})}{e^{-\lambda a}}, \forall \lambda > 0$ with the Chernov bound in [86].

$E(e^{-\lambda X}) = \sum_{k=0}^{N-1} P(X = k) e^{-\lambda k} = \sum_{k=0}^{N-1} \binom{n-1}{k} p^k (1-p)^{N-1-k} e^{-\lambda k} = (1 - p + pe^{-\lambda})^{N-1}$.

With $a = Np(1-\delta)$, $P(X < Np(1-\delta)) < \frac{E(e^{-\lambda X})}{e^{-\lambda a}} = \frac{(1-p+pe^{-\lambda})^{N-1}}{e^{-\lambda Np(1-\delta)}}$.

With $p = \frac{r^2}{L^2}$ and $N = \mu L^2$, $P(X < \mu r^2(1-\delta)) < (1+\frac{r^2}{L^2}(e^{-\lambda}-1))^{-1} e^{N \ln(1+\frac{r^2}{L^2}(e^{-\lambda}-1))} e^{\lambda(1-\delta)\mu r^2}$.

Using $\ln(1-x) < -x$, $P(X < \mu r^2(1-\delta)) < (1+\frac{r^2}{L^2}(e^{-\lambda}-1))^{-1} e^{(N-1)\frac{r^2}{L^2}(e^{-\lambda}-1)} e^{\lambda(1-\delta)\mu r^2}$.

By fixing $\lambda = \delta$, $P(X < \mu r^2(1-\delta)) < (1+\frac{r^2}{L^2}(e^{-\delta}-1))^{-1} e^{\mu r^2(e^{-\delta}-1+\delta-\delta^2)}$.

Using $e^{-\delta} < 1-\delta+\frac{\delta^2}{2}$, $P(X < \mu r^2(1-\delta)) < (1+\frac{r^2}{L^2}(e^{-\delta}-1))^{-1} e^{-\frac{\mu r^2}{2}\delta^2}$.

Then, finally we have $P(X < \mu r^2(1-\delta)) < (1-\frac{2r^2}{L^2})^{-1} e^{-\frac{\mu r^2}{2}\delta^2}$.

We choose r so that G fits exactly and any $r \times r$ square is not truncated. The exact fitting is achieved by taking the origin point of $\mathbb{R}^2$ as the center of the square G, and by selecting $r = \frac{2k+1}{L}$ where k is a positive integer. Hence, When $L \to \infty$ and $\mu \to \infty$ with $\delta > 0$, $(1-\frac{2r^2}{L^2})^{-1} e^{-\frac{\mu r^2}{2}\delta^2} \to 0$. Therefore, $\frac{m_{\min}}{\mu r^2} \xrightarrow{p} 1$

In addition, $P(X > a) = P(e^{\lambda X} > e^{\lambda a}) < \frac{E(e^{\lambda X})}{e^{\lambda a}}, \forall \lambda > 0$ with the Chernov bound in [86].

$E(e^{\lambda X}) = \sum_{k=0}^{N-1} P(X=k)e^{\lambda k} = \sum_{k=0}^{N-1} \binom{n-1}{k} p^k (1-p)^{N-1-k} e^{\lambda k} = (1-p+pe^{\lambda})^{N-1}$.

With $a = Np(1+\delta)$, $P(X > Np(1+\delta)) < \frac{E(e^{\lambda X})}{e^{\lambda a}} = \frac{(1-p+pe^{\lambda})^{N-1}}{e^{\lambda Np(1+\delta)}}$.

With $p = \frac{r^2}{L^2}$ and $N = \mu L^2$, $P(X > \mu r^2(1+\delta)) < (1+\frac{r^2}{L^2}(e^{\lambda}-1))^{-1} e^{N \ln(1+\frac{r^2}{L^2}(e^{\lambda}-1))} e^{-\lambda(1+\delta)\mu r^2}$.

Using $\ln(1-x) < -x$, $P(X > \mu r^2(1+\delta)) < (1+\frac{r^2}{L^2}(e^{\lambda}-1))^{-1} e^{(N-1)\frac{r^2}{L^2}(e^{\lambda}-1)} e^{-\lambda(1+\delta)\mu r^2}$.

By fixing $\lambda = \ln(1+\delta)$, $P(X > \mu r^2(1+\delta)) < (1+\frac{\delta r^2}{L^2})^{-1} e^{\mu r^2(\delta-(1+\delta)\ln(1+\delta))}$.

Then, we have $P(X > \mu r^2(1+\delta)) < (1+\frac{\delta r^2}{L^2})^{-1} e^{-\frac{\mu r^2}{2}(\delta^2+O(\delta^3))}$, and finally $P(X > \mu r^2(1+\delta)) < (1-\frac{2r^2}{L^2})^{-1} e^{-\frac{\mu r^2}{2}\delta^2}$.

When $L \to \infty$ and $\mu \to \infty$ with $\delta > 0$, $(1-\frac{2r^2}{L^2})^{-1} e^{-\frac{\mu r^2}{2}\delta^2} \to 0$. Therefore, $\frac{m_{\max}}{\mu r^2} \xrightarrow{p} 1$

Asymptotic Values of the Min-Cut of Unit-Disk Graphs

Theorem 8 *For a sequence of random unit disk graphs and associated sources $(\mathcal{V}_i, s_i \in \mathcal{V}_i)$, with fixed radio range ρ, fixed border area width W, with a size $L_i \to \infty$, and a density $M = L^\theta$ with fixed $\theta > 0$, we have the following limit of the min-cut $C_{\min}(s_i)$:*
$\frac{C_{\min}(s_i)}{M} \xrightarrow{p} 1$ in probability. Additionally : $\frac{M_{\max}}{M} \xrightarrow{p} 1$

Proof: With Theorem 7, we have $\frac{m_{\min}}{\mu r^2} \xrightarrow{p} 1$ and $\frac{m_{\max}}{\mu r^2} \xrightarrow{p} 1$ in probability, when $L \to \infty$ and $\mu \to \infty$.

Consider the bound of Theorem 6: the min-cut $C_{\min}(s)$ of the graph $\mathcal{V}$, verifies: $C_{\min}(s) \geq m_{\min}(|R(r)|-1)$, hence: $\frac{C_{\min}(s)}{M} \geq \frac{m_{\min}(|R(r)|-1)}{M}$

The right side of the inequality is:

$a = \frac{m_{\min}(|R(r)|-1)}{M} = \frac{m_{\min}}{\mu r^2}\frac{\mu}{M}r^2(|R(r)|-1)$

We have:

- $\frac{m_{\min}}{\mu r^2} \xrightarrow{p} 1$ in probability,
- $\frac{\mu}{M} = \frac{1}{\pi\rho^2}$
- $r^2(|R(r)|-1) = \pi\rho^2(1+O(\frac{1}{r}))$, from Lemma 7.

Therefore the right side $a \xrightarrow{p} 1$ in probability. This gives a lower bound of $\frac{C_{\min}}{M}$ for $L \to \infty$.

Let us show that this lower bound is also an upper bound (in probability). Recall that the min-cut $C_{\min}$ is lower than any cut, for instance one cut with only neighbors of a node t ($T = \{t\}$). Let us consider the node $t \in \mathcal{V}$ with the maximum number of neighbors $M_{\max}$, and hence $C_{\min} \leq M_{\max}$

We have: the maximum number of neighbors $M_{\max}$ is at most $m_{\max}|R^+(r)|$, where $R^+(r)$ is similar to $R(r)$, except considering squares of around a point of the lattice with radius $\rho + 2r$. Like for $|R(r)|$, one can prove:

$$R^+(r) = \pi\frac{\rho^2}{r^2}(1+O(\frac{1}{r}))$$

and like $m_{\min}$, one can show that $\frac{m_{\max}}{\mu r^2} \xrightarrow{p} 1$ in probability.

Collecting these properties, we get:

$$\frac{C_{\min}}{M} \leq \frac{M_{\max}}{M} \leq \frac{m_{\max}|R^+(r)|}{M}$$

where the right side of the bounds : is such that $\frac{m_{\max}|R^+(r)|}{M} \xrightarrow{p} 1$ in probability.

Hence, the upper bound, and the theorem.

4.6 Experimental Efficiency Evaluation

In section 4.5, we theoretically analyzed the efficiency of our approach. In this section, we experimentally evaluate our approach in practical networks. Considering that the expected number of neighbors is irregular in practical networks, we use IRMS as a rate selection method. The experimental study is done first in four different kinds of graphs: instances of lattice unit disk graphs and random unit disk graphs, both with and without torus effect. Then, more intensive experimental study is done in random unit disk graphs with different network scales and densities.

4.6.1 Efficiency in Different Types of Networks

In order to evaluate the efficiency of the heuristic IRMS, we first measured a relative efficiency that is a ratio of E_{cost} and $E_{optimal}$ like; $E_{\text{rel-eff}} \triangleq \frac{E_{\text{optimal}}}{E_{\text{cost}}}$. The relative efficiency is based on a comparison of E_{cost} from the heuristic IRMS and $E_{optimal}$ from the optimal solution as described in section 4.3.2.

One reference point is the achievable relative efficiency without network coding in [73]. In order to obtain the approximate upper bound, we use the same logic of $E^{(\text{no-coding})}_{\text{rel-cost}}$ detailed in section 4.3.2, except we invert it as $\frac{1}{E^{(\text{no-coding})}_{\text{rel-cost}}}$. The reason, for the inversion is that relative efficiency is obtained by dividing $E_{optimal}$ by E_{cost} (an inversion of $E^{(\text{no-coding})}_{\text{rel-cost}}$). Hence the approximative upper bound of efficiency without coding translates into: $E^{(\text{no-coding})}_{\text{bound-rel-eff}} \approx 0.609\ldots$.

For comparison purposes, we evaluate the IRMS heuristic on four different kinds of graphs: instances of lattice unit disk graphs and random unit disk graphs, both with and without torus effect. The parameters for all four variations are the following: the total node number $N = 196$, and the average neighbor number is successively $4, 12, 28, 48, 80$.

Figure 4.20 represents the results of relative efficiency from the variations and one reference point. We obtained the results in Figure 4.20 for ten different instances of each variation. The central result is the appreciable efficiency of IRMS compared to the achievable efficiency without network coding: it mostly outperforms the upper bound without network coding (0.609).

Results from the four different types of networks are detailed as:

- First type, lattice unit disk graphs with torus effect: These are the most regular graphs and in these graphs, IRMS closely approaches optimality (> 0.95). The high efficiency

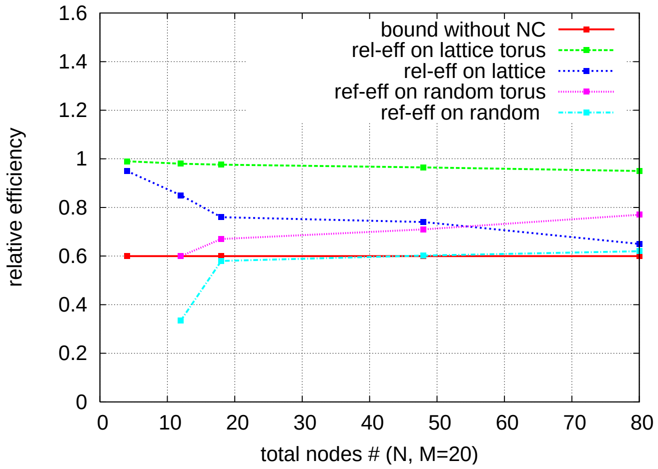


Figure 4.20: Relative cost (efficiency compared to optimal)

occurs because every node has the same number of neighbors without the border effect. Hence, the rate adjustment is not necessary.

• Second type, lattice unit disk graphs without torus effect: a node generally has the same number of neighbors, except for the nodes near the border. The border nodes with fewer neighbors can possibly reduce the efficiency and become problematic. However, IRMS successfully solves the border issue as Figure 4.20 shows: the relative-efficiency of IRMS approaches the targeted one $\frac{\text{min}-\text{cut}}{M} = 1$. On the other hand, the good efficiency slightly decreases as the density increases. The slight decrease occurs because many nodes generally have an overlapped transmission range and all nodes in the overlapped transmission range increase their rates for the worst case of starvation. Namely, a few extremely starving nodes can increase the rates of all nodes within a transmission range.

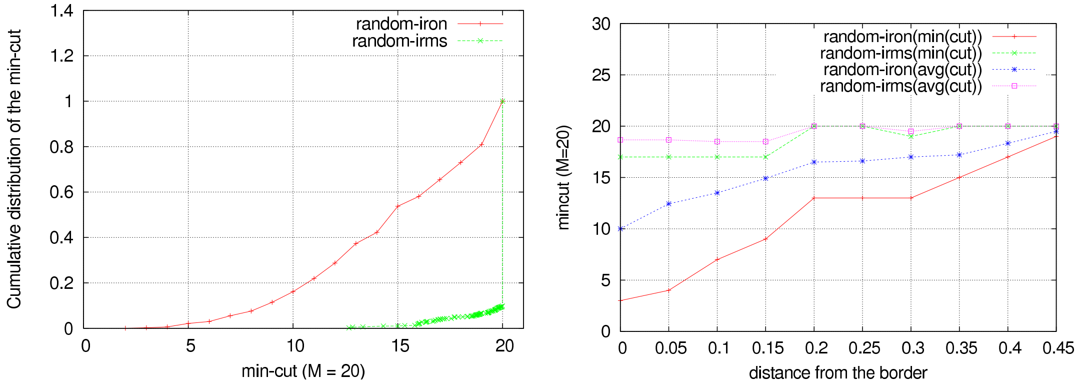
• Third type, random unit disk graphs with torus effect: No border effect here. However, IRMS has to overcome the effects of the irregular neighbor number; the graph of IRMS shows that IRMS convincingly overcomes it.

• Fourth type, genuine random unit disk graphs: these graphs are closest to real networks. The efficiency on the graphs is higher than the reference point without coding, and hence its efficiency is acceptable. The only noticeable problem is that efficiency becomes lower as density decreases. Because of the efficiency decrease, IRMS does not fully achieve the maximum broadcast rate, M in sparse networks. We detail the exceptional cases in section 4.6.4

4.6.2 Distribution of the Min-cut

In this section, we provide further results from the random unit disk graphs; these graphs were the ones with the most problematic efficiency among the graphs of four variations.

Remember that the minimum of the min-cuts $C_{\min}(s,t)$ (for all $t \in \mathcal{V}$) decides the overall maximum broadcast rate of the source. Hence, good efficiency is achieved when the distribution of these min-cuts is tighter and more concentrated. Deeper insight is gained by analyzing the cumulative distribution of the min-cut of each node for the random graph $N = 400$ previously studied. The cumulative distribution of the min-cut is displayed in Figure 4.21(a). As evidenced, without IRMS, the distribution of the min-cut is wider from 5 to 20, but with IR-MS, the distribution is closer to M (with a peak for $M = 20$, the targeted min-cut). The concentrated min-cuts around M showed that almost all nodes achieved the targeted min-cut M except a few nodes. However, the few nodes whose min-cut is around 15 demonstrate that there is still some room for improvement.



(a) mincut: cumulative distribution

(b) mincut vs distance from the border

Figure 4.21: Mincut distribution

Figure 4.21(b) provides additional information about the value of a min-cut depending on the position of the nodes, for IRMS and IRON respectively. The distance of each node to the border of the network was computed, and we performed a statistical analysis for the min-cuts of nodes that have the same distance to the border. Figure 4.21(b) gives the average min-cut and the minimum min-cut for nodes at a given distance from the border. In Figure 4.21(b), we can see that the nodes closer to the border generally have a lower min-cut. But, the lowest min-cut with IRMS is 90 percent of M. On the other hand, the lowest min-cut without IRMS is much smaller, only 15 percent of M. Consequently, Figure 4.21(b) shows that the border effect is a key to producing a lower min-cut; again, we see that

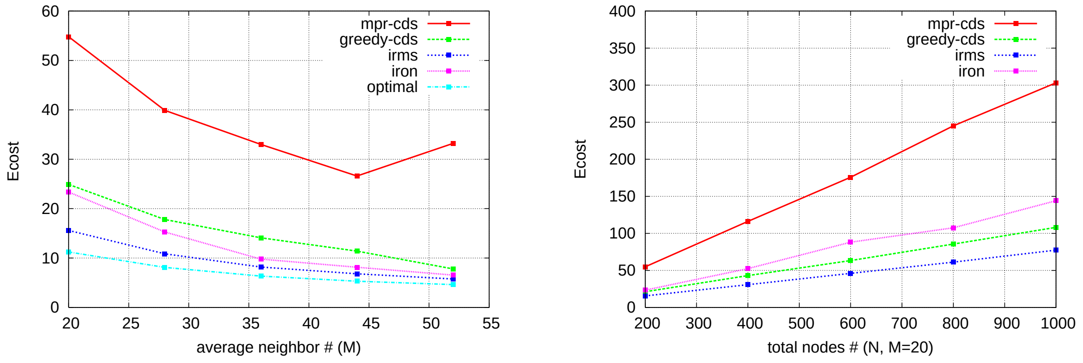
IRMS improves the minimum min-cut but does not always achieve the target min-cut$= M$.

4.6.3 Random Unit Disk Graphs N, M

Figure 4.23 and 4.22 show different perspectives on the efficiency with and without torus effect. In order to measure the efficiency on these graphs, we use E_{cost} which indicates the number of retransmissions per broadcast packet, as a metric.

First, different algorithms are compared:

- Network coding with IRON, which does not consider the border effect.
- Network coding with IRMS which is based on the static adaptation .
- MPR-based dominating sets from [5] (with efficiency close to MultiPoint-Relays (MPR)-based flooding of [6]). It is representative of the efficiency of algorithms with only local topology information based on store-and-forward routing.
- Connected dominating sets from [4] (efficient variant of a greedy algorithm). It is an efficiency representative of centralized connected dominating set algorithms (and to some extent, representative of what could be achieved without network coding).
- The optimal solution with network coding obtained by solving a linear optimization problem for



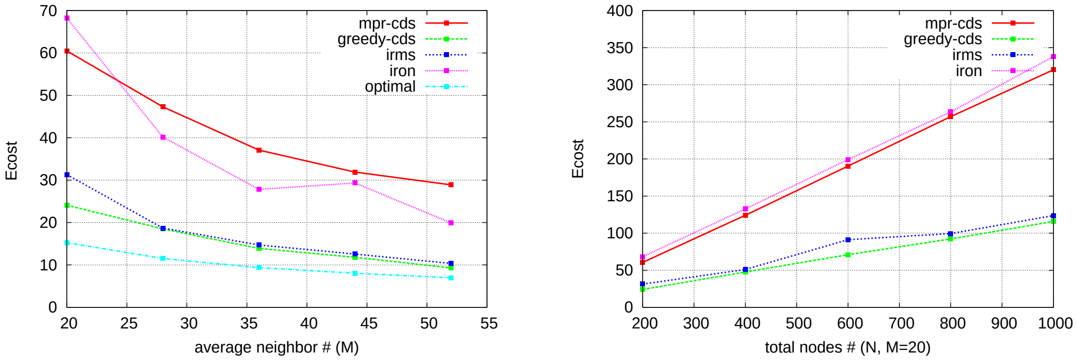
(a) cost vs average neighbor number N=200

(b) cost vs total node number M=20

Figure 4.22: E_{cost} on random unit disk graphs with torus effect

Figure 4.22(a) displays the efficiency of each algorithm with density increase on random networks with torus effect. One result is that the efficiency of an optimal solution with network coding is about $\frac{1}{3}$ better than the efficiency of the connected dominating set using precise total topology from [4] with store-and-forward routing. In addition, the relative efficiencies of IRMS and IRON are also similar to the efficiency of the optimal solution.

In the case of IRMS, the efficiency gap with the optimal solution is less than 25 percent. Moreover, the efficiency of IRMS and IRON is much higher than the efficiency of MPR-based broadcasting. The low efficiency of MPR-based broadcasting shows the difficulty of achieving high efficiency only using local information. We also find that there exists a small gap between the relative efficiencies of IRMS and IRON. Even if the gap is small, it shows an irregularity in neighborhood size away from the border. IRMS successfully resolves the efficiency decrease that occurs because of this irregularity as seen in Figure 4.22. In addition, Figure 4.22(b) confirms that the heuristic IRMS maintains its efficiency even if a network becomes larger.



(a) cost vs average neighbor number on square N=200

(b) cost vs total node number on square M=20

Figure 4.23: E_{cost} on random unit disk graphs without torus effect

Figure 4.23(a) demonstrates the efficiency of each algorithm on the same graphs, but without torus effect. Similar to results on networks with torus effect, the efficiency of the optimal network coding is about $\frac{1}{3}$ better than the efficiency of the connected dominating set with precise global topology from [4] without store-and-forward routing. However, we can see that the efficiency of IRON dramatically decreases in networks without torus effect. The dramatic decrease again confirms the importance of addressing the border effects in static networks. Unlike IRON, IRMS maintains a similar efficiency to that of the optimal solution with network coding on networks without torus effect. Moreover, IRMS is as efficient as CDS based broadcasting using precise global topology although IRMS only uses simple local topology information, the number of neighbors. The good performance of IRMS is already an interesting result. However, even if the gap is small, there exists an efficiency gap between results from IRMS and the optimal. The small gap again shows the difficulty of achieving high efficiency just using only local information. Lastly, we can confirm again that IRMS keeps its high efficiency independently with network scales.

4.6.4 Difficulties for Distributed Rate Selection

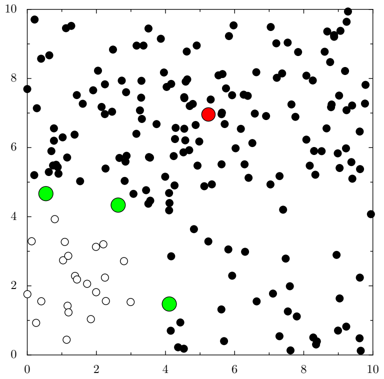


Figure 4.24: Example of cut

Finally, Figure 4.24 gives an example of an instance of a random unit disk graph, where the efficiency of IRMS was found to be low. The cut (S/T corresponding to the min-cut) for a node at the bottom-left corner is represented. The white dots represent nodes in T and black dots are nodes in S. The red dot represents a source and the green dots are the only nodes connecting the part in the corner, T to the rest of the network, S. Namely, the green dots represent nodes in ΔS in equation 3.4

With IRMS, the green nodes do not have extreme starvation in their neighborhood, and therefore increase their rate by only a small amount. But data from the source is only transmitted through these green nodes to the sets of nodes in T (white dots), and for this reason the green nodes' rates should be greater. This example perfectly illustrates the difficulties found in sparse networks: notice that coordination between the green nodes would require multi-hop communication and global topology in order to detect this issue.

4.7 Rate Selection Improvement : Dynamic Adaptation

In this section, we improve our rate selection algorithm so that it can be dynamically adapted to various networks, specifically to sparse networks, including those that are partitioned into several parts and are connected by only few nodes.

Our simple rate selection that essentially set the same rate on almost all nodes was theoretically shown to be asymptotically optimal. In addition, experimental results showed that our approach generally outperformed other broadcasting without network coding, and got closer to the optimal obtained by solving a linear optimization problem. However, in special cases such as sparse networks, it was found that the efficiency decreased and local topology information was not sufficient to detect such special cases.

In order to detect the special cases and improve the efficiency of the special instances of networks, we suggest a rate selection based on dynamic adaptation. The study of the dynamic adaptation started from the same general idea of finding not necessarily an optimal rate selection, but a simple and efficient rate selection. We also started from our basic rationale in section 4.4, but with the novel approach of using simple dynamic information to adapt the rates, rather than static topology information. The rate of each node is set from the current state of information inside the network. There are several advantages in doing so; first, the problematic cases that cannot be detected from static local topology information in section 4.6.4, may be discovered from dynamic information. Second, in real networks, the network itself evolves dynamically with packet loss rates or topology changes and hence the rate selection can no longer be a fixed constant rate. These dynamic features require dynamic adaptation in any case. To handle this issue, we propose a novel algorithm based on a dynamic rate selection method, especially in considering the initial rate. The key contributions of this chapter are the proposal of a dynamic heuristic for rate selections, its analysis using insights from theory, and an experimental investigation of the performance with simulations.

The rest of this section is organized as follows: section 4.7.1 details the new heuristic, DRAGON, section 4.7.2 analyzes the performance with experimental results and section 4.7.2.2 concludes this chapter.

4.7.1 Dynamic Heuristic for Rate Selection

In this section, we introduce our improved heuristic with dynamic adaptation. Before introducing the heuristic, we first explain the general concept of *cut at destination* and *local received rate* in section 4.7.1. These concepts are connected to the maximum broadcast rate that is theoretically a min-cut as detailed in section 3.3.3.

Local Received Rate: Cut At Destination

The cut at destination is a special case of a cut as defined in section 3.3.3. Formally, it is the partition S/T of the nodes, where T includes only the destination node $T = \{t\}$ (hence $S = \mathcal{V} \setminus \{t\}$). An example is represented in Figure 4.25: the node t has neighbor nodes v_1, v_2, v_3 and v_4 (this is the set ΔS of the cut in section 3.3.3); the capacity of the cut is the sum of the rate of these nodes, $C(S) = C_{v_1} + C_{v_2} + C_{v_3} + C_{v_4}$ (as shown by (3.4) in section 3.3.3).

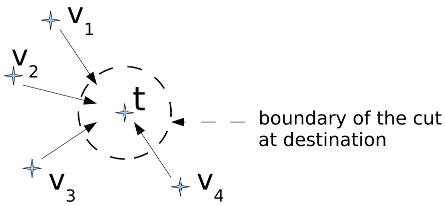


Figure 4.25: Cut S/T at destination: $\Delta S = \{v_1, v_2, v_3, v_4\}; T = \{t\}$

The capacity of the cut $C(S)$ may be pictured as the total rate that flows through the boundary between S and T, and the capacity of the cut at the destination is simply the total rate at which one node receives innovative packets from its neighbors: the *local received rate*.

Cut at Destination, Min-Cut and Efficiency From section 3.3.3, the maximum broadcast rate of the source is the capacity of the min-cut. Therefore, it is lower than the capacity of any cut at destination (= any local received rate).

Notice that in general, the min-cut need not be the cut at one destination: for instance in the example topology of Figure 4.26, one unique node is connecting the left and right parts of the network. If, for instance, all the nodes had an identical average rate lower than the source, then that center node would be the bottleneck, and the min-cut would be the partition S/T with the following features: on one side, all nodes in the right part, which

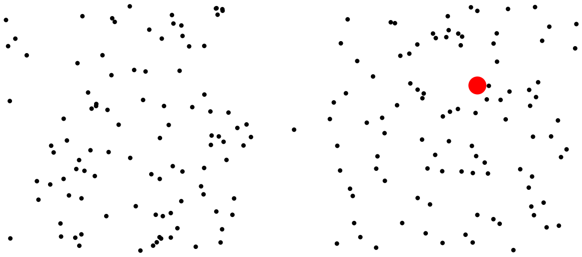


Figure 4.26: Example topology (source is the larger node)

contains the source plus the center (set S); and on the other side, all the nodes of the left part (set T).

One link between the cut at destination and a min-cut comes from our proof in section 4.5.2 and [85]. One result of the proof is that the min-cut is essentially equal to the number of neighbors; this was proved for lattice networks and for random unit-disk networks as the density grows towards infinity. This implies that in those cases, the min-cut is essentially one cut at destination [2].

In addition, under the same asymptotic conditions, it was proved that on average the proportion of non-innovative transmissions would converge towards 0. This proved asymptotically optimal efficiency for the metric of the number of transmissions per broadcast packet, and demonstrated an advantage over non-coding.

Static Heuristics using Cut At Destination As seen in the previous section, the min-cut is related to the local received rate in some cases of homogeneous networks. However, when the nodes have a different number of neighbors, they have different local receiving rates. The first step is to adjust the different local receiving rates, in order that the local receiving rate would be at least the broadcast rate of the source, M, for every destination. Indeed, every node in the network should receive at least a total rate M from its neighbors.

We explored our simple heuristic to achieve the same local receiving rate M at every node in section 4.4 by following the simple logic: when a node has h neighbors, each of its neighbors would have a rate at least equal to $\frac{M}{h}$. Then the local receiving rate is always at least the sum of h such rates, indeed greater or equal to M. This yields our static heuristic, IRMS in section 4.4.

In [73], and in a slightly different context (not explicitly a rate selection, broadcast

[2] or at the that the min-cut as a similar capacity to the cut at destination

all-to-all), theoretical arguments were given for the ability to decode in the case of general networks when $k \geq 3$. But [85] shows that $k = 1$ is asymptotically sufficient for one-to-all broadcasting.

We also experimentally explored the performance of IRMS ($k = 1$) in section 4.5. Although overall good performance was observed, in the case of sparser networks, phenomena occurred where only a few nodes would connect one part of the network to another, in a similar fashion to the center node in Figure 4.26. In networks similar to Figure 4.26, the rate of the nodes linking two parts of the network would be dependent on how many of them are present. Such information is not available from local topology information.

Dynamic Heuristic, DRAGON Our heuristic IRMS for rate selection is static, using simple local topology information, and the rates would be constant as long as the topology remained identical.

However, a different approach is possible. The starting point is the observation that with fixed rates, the rank of a node v, will grow linearly with time, as $D_v(\tau) \approx C_{\min}(s, v) \times \tau$, i.e. proportionally to the capacity of the min-cut from the source s to the node (see section 3.3.3). With a correct rate selection, one would expect this min-cut to be close to the source rate $C_{\min}(s, v) \approx M$ in every node, and would therefore expect that the ranks of all nodes would grow at the same pace. Failure of the ranks to grow at the same pace is a symptom of the fact that the rate selection requires adjustment.

From $D_v(\tau)$, the rank of a node v at time τ, let us define $g_v(\tau)$, the maximum gap of rank with its neighbors, normalized by the number of neighbors, that is:

$$g_v(\tau) \triangleq \max_{u \in H_v} \frac{D_v(\tau) - D_u(\tau)}{|H_u|}$$

We propose the following rate selection, DRAGON *Dynamic Rate Adaptation from Gap with Other Nodes*, which adjusts the rates dynamically, based on the gap in the relative ranks of a node and its neighbors as follows:

- DRAGON: the rate of node v is set to $C_v(\tau)$ at time τ as:
  - if $g_v(\tau) > 0$ then: $C_v(\tau) = \alpha g_v(\tau)$
    where α is a positive constant
  - Otherwise, the node stops sending encoded packets until $g_v(\tau)$ becomes larger than 0

This heuristic has some strong similarities, with the reasoning presented in section 4.7.1, and with IRMS. Consider the cut at one destination v. In this case, DRAGON ensures that every node will receive a total rate at least equal to the average gap of one node and its neighbors scaled by α, which as the local received rate at time τ verifies:

$$C(\mathcal{V} \setminus \{v\}) \geq \alpha \left(\frac{1}{|H_v|} \sum_{u \in H_v} D_u(\tau) - D_v(\tau) \right)$$

This would ensure that the gap would be closed in time $\approx\leq \frac{1}{\alpha}$, if the neighbors did not receive new innovative packets.

Theoretical Analysis of DRAGON

Overview In effect, DRAGON is performing a feedback control, which intends to equalize the rank of one node to the rank of its neighbors, by adapting the rates of the nodes.

However, notice that the precise control-theoretic analysis of DRAGON is complicated by the fact that the rank gap does not behave like a simple “physical” output, and that we have the following properties, for two neighbor nodes u, v:

- If $D_u > D_v$, then every transmission of u received by v will increase the rank of v (is innovative).
- If $D_u \leq D_v$, then a transmission of u received by v, may or may not, increase the rank of v. Both cases may occur.

The last property may be illustrated in the network in Figure 4.26. Consider the center node: it is the only node connecting the two parts of the network; hence all nodes on the left part of the network received vectors that went through it, and any transmission received from the left, will not be innovative for the center node. Now, imagine that another node is added, out of range of the center node, such that it also connects the two parts of the network. In that case, some transmissions from the left are likely to become innovative for the center node (depending on the overall rate selection).

As a result, there is some uncertainty in the control regarding the effect of increasing the rate of a neighbor that has greater rank than another node. A refined dynamic approach would require gathering detailed statistics about the innovation rate, and using this information in the control; however, the approach used in DRAGON is simpler and more direct. If a node has a lower rank than all its neighbors, it will stop sending packets. This fact causes us to pessimistically estimate that transmissions from nodes with lower rank are non-innovative.

Although this would tend to make the rates less stable with time, intuitively this property might allow DRAGON to be more efficient. Note also from section 3.3.3, that there is no direct penalty for unstable rates: only the average rates matter for the maximum broadcast rate (and also for the cost).

Insights from Tandem Networks Although the exact modeling of the control is complex, some insight may be gained from approximation.

Consider one path ($s = v_0, v_1, \dots v_n$) from the source s to a given node v_n, such as in Figure 4.27.

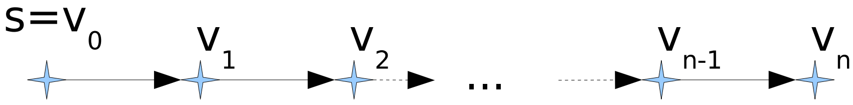


Figure 4.27: Line network

Denote $D_k \triangleq D_{v_k}$, $C_k \triangleq C_{v_k}$. Assume now that the ranks in the nodes verify $D_{k+1}(\tau) < D_k(\tau)$, for $k = 0, \dots, n-1$, and then a fluid approximation yields the following equations:

$$\frac{dD_{k+1}(\tau)}{d\tau} = C_k(\tau) + I_k(\tau) \text{ with } \alpha_k = \frac{\alpha}{|H_{v_{k+1}}|} \tag{4.15}$$

$$C_k(\tau) = \alpha_k(D_k(\tau) - D_{k+1}(\tau - \delta_k(\tau))) \tag{4.16}$$

where $I_k(\tau)$ is the extra innovative packet rate at v_k from neighbors other than v_{k-1} and $\delta_k(\tau)$ is the delay for node $k+1$ to get information about the rank of its neighbor k. If the rank is piggybacked on each transmitted packet, then $\delta_k(\tau) \approx \frac{1}{C_k(\tau)}$.

If we make the approximation that $\delta_k(\tau) = 0$, and if we consider a linear network composed exclusively of the single path, then $\alpha_k = \frac{1}{2}\alpha$ and $I_k(\tau) = 0$. Let $\beta \triangleq \frac{1}{2}\alpha$. In that case, (4.15) and (4.16) yields the following sequence of first order equations for fixed source rate M and $D_k(0) = 0$.

$$\frac{dD_{k+1}(\tau)}{d\tau} = \beta(D_k(\tau) - D_{k+1}(\tau))) \tag{4.17}$$

In this case, (4.15) and (4.16) yields a sequence of first order equations for D_k, solvable with standard resolution methods:

$$D_k(\tau) = M\tau - (1 - e^{-\beta\tau})\frac{kM}{\beta} - e^{-\beta\tau} P_k(\tau) \tag{4.18}$$

where $P_k(\tau)$ is a polynomial of τ with $P_k(0) = 0$. This result shows that in this case the ranks are $D_k(t) \approx M\tau$ plus an additional term: when $\tau \ll \frac{1}{\beta}$, this term is $\approx P_k(\tau)$ and is 0 for $\tau = 0$; whereas when $\tau \gg \frac{1}{\beta}$ this term is $\approx \frac{kM}{\beta}$.

The conclusion is that, for a line network, when $\tau \rightarrow \infty$, the rank of the nodes $v_0, v_1, \ldots, v_k$ are such that the dimension gap between two neighbors is $\frac{M}{\beta}$, and this occurs after a time on the order of magnitude of $\frac{1}{\beta} = \frac{2}{\alpha}$: the ranks of the nodes decrease linearly from the source to the edge of the network.

4.7.2 Experimental Results

In order to evaluate the performance of DRAGON, we performed extensive simulations, which are detailed in this section. The focus of the DRAGON algorithm is on wireless ad hoc networks, and simulations were performed either for random uniform graphs (inside a square) or with the reference network of Figure 4.26, in which one node connects two parts. This last network is of special interest because it exemplifies features found in sparse networks in Figure 4.24 in section 4.6.4, where static heuristics fail.

Parameter	Value(s)
Number of nodes	200
Range	defined by M, expectation of number of nodes in one neighborhood. $M = 8$
Position of the nodes	random uniform i.i.d, a network on Figure 4.26
Generation size	500
Field $\mathbb{F}_p$	$p = 1078071557$
α (in DRAGON)	$\alpha = 1$

Table 4.1: Default simulation parameters.

The default simulation parameters are given in Table 4.1. The simulator used was self-developed, with an ideal wireless model with no contention, no collision and instant transmission/reception. For comparison purposes, NS-2 (version 2.31) was also used with its default parameters.

The metric for (energy-)efficiency used in these simulations is: the total number of transmitted packets per source packet, and is denoted as E_{cost}. $E_{\text{cost}} \triangleq \frac{\text{Total number of transmitted packets}}{\text{Generation size}}$. As mentioned in section 4.3.2, the optimal rate selection may be computed from a linear program [13]: it is the rate selection that minimizes E_{cost}. In some scenarios, we computed numerically this minimum E_{cost} (by a linear program solver), to obtain a reference point.

We also implemented a termination protocol which decides whether to stop transmission based on the state of neighbors, and chose the generation size to be sufficiently large to offset its cost.

General Behavior of Wireless Network Coding A first general view of the behavior of wireless broadcast with network coding would separate the broadcast into three durations: the starting phase, the general phase, and the termination phase. We simulated IRMS with default parameters on the network in Figure 4.26 and the simulation result is illustrated in Figure 4.28: the average innovative packet rate received by nodes in the network, depending on the time. In the starting phase, the nodes do not have enough vectors, and then the efficiency of network coding is limited. Hence the innovative rate is limited. In the general phase, network coding acts as predicted. In the termination phase, some nodes have all the packets (and are able to decode them), but some others do not. When the generation size is sufficiently large, the performance is dominated by the behavior in the central part.

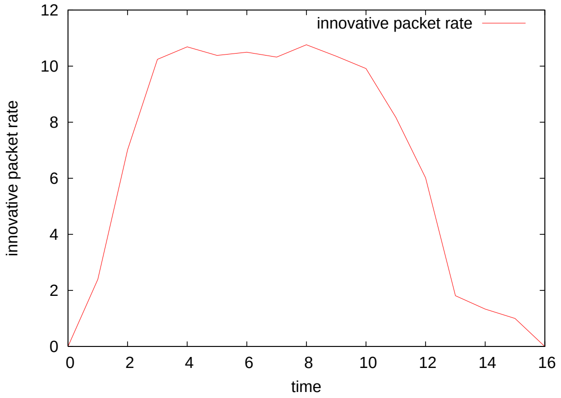


Figure 4.28: Average innovative packet rate vs time

Efficiency of DRAGON In this section, we start the analysis of the performance of various heuristics, by considering their efficiency from E_{cost}. Simulations were performed on several graphs with default simulation parameters except for the density (the density is sparse and hence the expected number of neighbors is 6) using three rate selections: optimal rate selection, IRMS, and DRAGON.

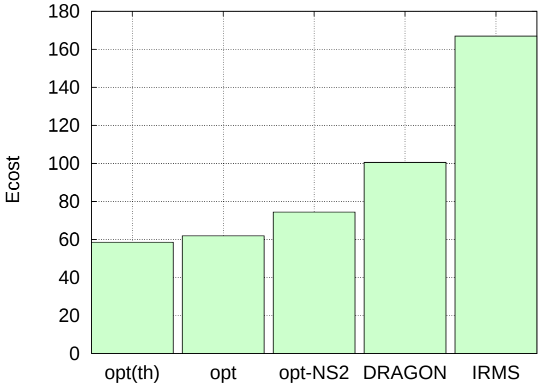


Figure 4.29: Performance of different heuristics

Figure 4.29 represents the results (for an average of 6 random graphs, M=6).

Theory and Practice The first 3 bars, are unrelated to DRAGON, and in essence, attempt to capture the gap between theory and practice.

The first bar (label: $opt(th)$), is the optimal E_{cost} as obtained directly from the linear program solution, without simulation. The second bar (label: opt), is the actual measured E_{cost} in simulations in the ideal wireless model, with optimal rate selection. The third bar (label: $opt - NS2$) is the actual measured E_{cost} in simulations using the simulator $NS - 2$ with parameters in Table 4.2.

As one might see, with the default parameters, the measured efficiency when performing actual coding with optimal rates (and generation size = 500), gives a result rather close to the numerical value of the optimal within a few percents. Another result is that the impact of the physical and MAC layer, as simulated by NS-2 with parameters in Table 4.2, is limited as one can see: $\approx 20\%$ (and the channel occupation rate was approximately 1/3).

The results are close. Therefore, because the purpose of our algorithm is not to perform congestion control (and because the parameters chosen would make some simulations operate above the channel capacity), in the remaining results, we will present results with the ideal wireless model.

Parameter	Value(s)
Number of nodes	200
Transmission range	250m
Network field size	2300m x 2300m
Antenna	Omni-antenna
Propagation model	Two-way ground
MAC	802.11
Generation size	500
Field $\mathbb{F}_p$	$p = 1078071557$

Table 4.2: Default simulation parameters of NS-2

Efficiency Improvement for Sparse Networks The two last bars of Figure 4.29 represent the efficiency of DRAGON and IRMS respectively. As one may see in this scenario, the ratio between the optimal rate selection, and DRAGON is around 1.6, but without reaching this absolute optimum, DRAGON still offers a significantly superior performance to IRMS.

The gain in performance comes from the fact that the rate selection IRMS, has a lower maximum broadcast rate (in some parts of the network) than the actual targeted one, and therefore also lower than the actual source rate. As a result, in the parts with lower min-cut, the rate of the nodes is too high compared to the innovation rate whereas with DRAGON such phenomena should not occur for prolonged durations: This is one reason for its greater performance.

Field $\mathbb{F}_p$	p=2	p=23	p=17333	p=1078071557
Result	8302/200	8150/200	8127/200	8127/200

Table 4.3: Impact of field size

Impact of Field Size As indicated in section 3.3.3, the asymptotic performance is not related with field size. Table 4.3 shows the cost-per-broadcast E_{cost} for the reference graph in Figure 4.26, generation size = 200, and DRAGON. It appears that indeed there is little variation (2%). To be complete, notice that because DRAGON attempts to equalize the rank gaps, it is countering the effect that made field size secondary in the queueing model of [19]; hence the performance of DRAGON with smaller field size is expectedly different and slightly less efficient

Impact of Loss Rate By nature, random linear network coding has the property of being resilient to losses, in the sense that one loss may be compensated by just one additional transmission. Table 4.7.2 indicates the performance of DRAGON, in a lossless network and in a network with loss equal to 10%. The result is again the cost-per-broadcast E_{cost} (avg. number of transmissions to broadcast one source packet). As one can compute, in the case of the lossy network, the E_{cost} multiplied by the transmission success rate of the network (0.9) gives a value of 32.0364. This is a normalization of the results in order to factor out the loss rate. Then the normalized result with loss is only 0.15% higher than without loss: indeed network coding and DRAGON are highly resilient to loss in this example.

Loss Rate	0 %	10 %
Result	31.99	35.596

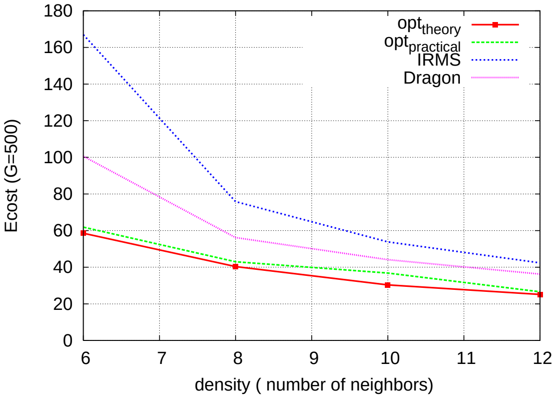


Figure 4.30: Increasing density

Impact of Density In Figure 4.30, simulations were performed on random graphs (avg. of 6), with default parameters and with increasing radio range. The modified parameter is M, with the average number of nodes in one disk representing radio range. As one may see, DRAGON performs well compared to IRMS in sparser networks, which are the most problematic cases, for reasons explained previously. For denser networks, the gap between IRMS, DRAGON and the optimal rate selection closes.

Figure 4.31 represents the impact of generation size: the efficiencies of the optimal rate selection (theoretical and with simulations), IRMS, and DRAGON are represented. As one might see, for the range of generation size selected, little variation is observed.

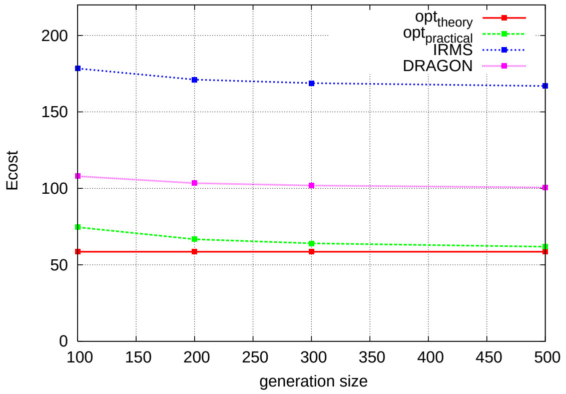


Figure 4.31: Impact of generation size

4.7.2.1 Closer Analysis of the Behavior of DRAGON

Impact of α In DRAGON, one parameter of the adaptation is α, and is connected to the speed at which the rates adapt. Table 4.7.2.1 indicates the total number of transmissions made, for the reference graph in Figure 4.26, for DRAGON with different values of α; and also for IRMS.

Value of α	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	IRMS
Total cost	16083	16272	17734	64411

Table 4.4: Impact of α

As one might see, first, the efficiency of IRMS on this network is rather low (1/4 of DRAGON). Indeed, the topology exemplifies properties found in the cases of networks where IRMS was found to be less efficient: two parts connected by one unique node. For various choices of α, it appears that the performance of *DRAGON* decreases when α increases. This evidences the usual tradeoff between speed of adaptation and performance. However the decrease in performance is rather limited, which we ascribe to two factors. The first one is, again, that for identical average rates, different rate selections will have asymptotically identical performance, whether the rates are oscillating dramatically or not. The second one is that in DRAGON, the nodes stop sending encoded packets whenever they are unsure whether their transmissions are beneficial to their neighbors.

Comparison with Model In Figure 4.32, Figure 4.33,Figure 4.34, X axis presents the distance from a source when a network has a 1 x 1 field. In Figure 4.32, Y axis presents

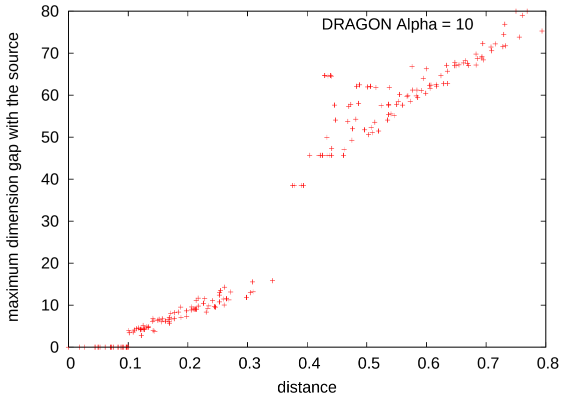


Figure 4.32: Maximum rank gap with the source

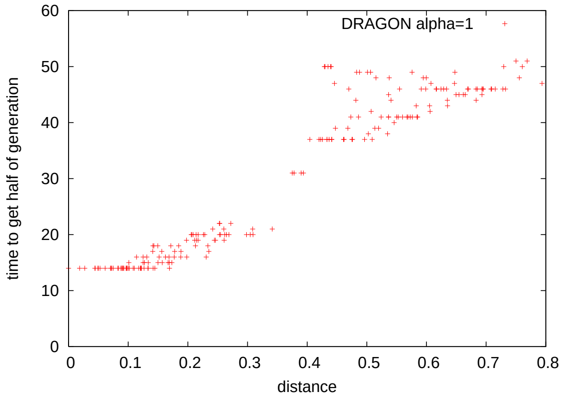


Figure 4.33: Propagation with distance $\alpha = 1$

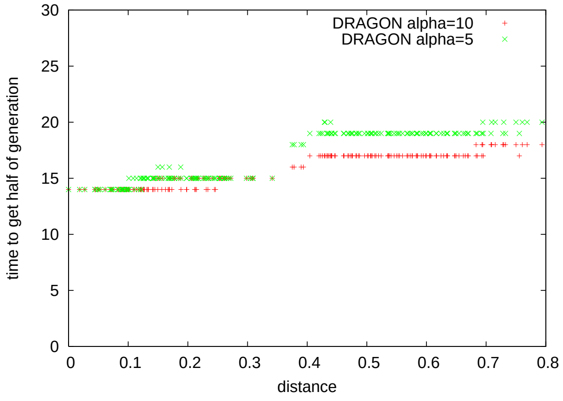


Figure 4.34: Propagation with distance $\alpha = 5, \alpha = 10$

dimension, the number of innovative packets, and In Figure 4.33 and Figure 4.34, Y axis presents simulation tick.

In Figure 4.32, some results are represented when running DRAGON with $\alpha = 10$ on the reference network shown in Figure 4.26. Consider one node. At every time, it has a rank, and this rank can be compared with the number of packets already sent by the source. The difference between the two should be ideally 0, and a larger value is an indication of the delay in propagation of the source packets. Hence that difference can be taken as a metric. We measure the following statistics: for each node, we identify the maximum value of that difference over the entire simulation. We then plot one point on the graph: the x coordinate is the distance of the node to the source, whereas the y is this difference.

Therefore the graph indicates how the "gap of rank with the source" evolves with distance. First, we see that there is a large step near the middle of the graph; this is the effect of the center node, which is the bottleneck and which obviously induces further delay. Second, two linear parts are present on each side of the step. This confirms the intuitions shown by the models in section 4.7.1 about a linear decrease in nodes' ranks from the source to the edge of the network.

Finally, one may find simulation results for $\alpha = 1$ in Figure 4.33 and for $\alpha = 5, \alpha = 10$ in Figure 4.34.

As in previous example, one dot represents a node in the network, and the x coordinate is the distance of the node to the source ; but this time the y-coordinate is the time at which the node has received exactly half of the generation size.

This yields further indication of the propagation of the coded packets from the source. Indeed if we compare the difference of time between nearest node from the source, and furthest node, we get a propagation time. It is around 35 for $\alpha = 1$, 6 for $\alpha = 5$ and 4 for $\alpha = 10$: roughly, it is inversely proportional to α, as expected from section 4.7.1 and seen in Figure 4.33 and Figure 4.34. In addition, the impact of the bottleneck at the center node is dramatically reduced as nodes on a large scale react to gaps with large α).

4.7.2.2 Summary of Dynamic Rate Selection for Sparse Networks

In this section, we introduced a dynamic heuristic for selecting rates: DRAGON. It did not assume a specific type of network topology; the only assumption was that one transmission reached several neighbors at the same time. The rate selection of DRAGON is constructed in a similar manner with our basic approach and static heuristic IRMS. As a result, the property of efficiency of DRAGON is inherited from our basic approach and static heuristic IRMS. Moreover, DRAGON also behaves as a feedback control: DRAGON detects nodes with inadequate rates using the expected dynamic behavior of network coding (namely, a linear increases with time of the information received), and accordingly adjusts the inadequacies. Because of this adaptation ability, DRAGON can use a pessimistic strategy, which decreases the rates of nodes whose transmissions are not absolutely guaranteed to be beneficial to their neighbors.

We experimentally explored the efficiency of DRAGON with various parameters including an error rate, an encoding field size, a generation size and network density. The experimental results show the effective feedback control of DRAGON, specifically in sparse networks including almost partitioned networks where partitioned parts are connected only by a few nodes. For instance, in some graphs of sparse networks, E_{cost} of DRAGON is just 58 percent of the cost of IRMS. Moreover, in the graphs of sparse networks, DRAGON achieves 70 percent of the theoretical optimal efficiency obtained by solving a linear optimization problem and 85 percent of the experimental optimal efficiency obtained by executing simulations with selected rates of the optimal solution.

4.8 Summary

In this chapter, we focus on using network coding as an *efficient* method to broadcast data to the entire wireless network, where the cost is the total number of transmissions. For single source broadcasting, results from information theory indicate that a maximum broadcast capacity may be achieved by a simple coding method, random linear coding [11, 19]. Then the key parameter for a given network is essentially the average retransmission rate of the nodes. As a result, finding an optimal solution to minimize the cost consists of finding the optimal average rate of every node: rate selection [13]. In fact, [12, 13, 67] have shown that optimal rate selection may be effectively obtained by solving linear programs in a polynomial time and possibly in a distributed fashion.

However, a different, even simpler, approach is possible. The approach is based on intuition about the condition to achieve maximal-efficiency-benefit of wireless network coding. The condition is to let every transmission bring new information to every receiver. Let us assume that the condition is satisfied and every node has the same number of neighbors. Then, every node could receive the same amount of innovative data per unit time if they select their transmission rate using IRON, which assigns the Identical transmission Rate one to most Other Nodes and the intended maximum broadcast rate (i.e. the expected number of neighbors, M) to a source.

The efficiency of the simple rate selection is theoretically explored in one dimensional torus networks and in two dimensional networks on Euclidean spaces. For the theoretical study, IRON is used as a rate selection method in one dimensional tours networks, and IRON is extended to IREN/IRON in two dimensional networks in order to handle border effects where nodes around the border definitely have smaller neighborhoods than others. With these rate selections, the theoretical study proceeds by comparing the efficiency of the rate selections with $E_{bound} = \frac{N}{M_{max}}$ (N: total node number, M_{max}: maximum neighbor number); E_{bound} is a bound of a possible minimum efficiency for wireless network coding. The comparison results converge to one in both one dimensional torus networks and in two dimensional networks. These results show that our approach is asymptotically optimal and our broadcasting with network coding is at least as efficient as other broadcasting with store-and-forward routing.

Subsequently, we experimentally evaluate our approach by comparing it with other broadcast protocols with store-and-forward routing. The experimental comparison is done in practical networks where irregularity in neighbor number exists and the irregularity lets some nodes have lower receiving rates than an intended receiving rate (a max flow) to achieve the maximal-efficiency-benefit of wireless network coding. In order to solve the starvation of the receiving rate with low cost, we propose a simple heuristic based on local topology information that *Increases Rate for Most Starving node, IRMS.* In some experiments, IRMS shows that it could outperform an MPR-based broadcast protocol with local topology and could be as efficient as a CDS-based broadcast protocol using a greedy algorithm with global topology. The experimental results also show that the acceptable efficiency is maintained regardless of a network scale. But, in special cases such as sparse networks, we find the efficiency decreased and local topology information is not sufficient to detect such cases.

In order to detect these special cases and prevent efficiency decrease, we introduce a dynamic rate selection: Dynamic Rate Adaptation from Gap with Other Node (DRAGON) method. DRAGON operated as a feedback control, whose target is to equalize the amount of information in the same neighborhood. The target is based on the optimality condition, which is that the receiving rate is the same at each node as the expected number of neighbors. The same receiving rate could enable each node to have the same amount of innovative data if propagation delay is ignored. When the propagation is taken into account, at least nodes in the same neighborhood have a similar amount of innovative data. Observation in tandem networks also confirms this hypothesis by showing that new information moves from a source toward the end of a network and a node has less information as its position gets closer to the end of the network. Hence, neighbors should have similar amounts of innovative data if their distance from the source is similar to that of nodes in the same neighborhood. If neighbors have a considerable gap in their innovative data, the gap should be narrowed. To narrow it, DRAGON lets each node adjust its rate proportional to the largest positive gap with its neighbors; the positive gap means that a node has more innovative data than its neighbors. In order to experimentally evaluate the efficiency of DRAGON, we execute simulations with various parameters including different error rates, encoding field sizes, generation sizes and network densities. The simulation results show in some representative networks that E_{cost} of DRAGON is just 58 percent of the cost of IRMS in sparse networks and DRAGON could achieve 85 percent of the optimal efficiency obtained by executing simulations with an optimal solution.

In summary, we propose a different and simpler approach for efficient broadcasting with network coding instead of solving a linear optimization problem. Our simple broadcasting with network coding is originated from investigation of the maximal-efficiency-benefit of wireless network coding. We ask the following questions: what is the maximal-efficiency-benefit of wireless network coding, how can we achieve the maximal-efficiency-benefit, and can a practical and simple protocol achieve it? From the investigation, we develop a basic rationale to achieve the maximal-efficiency-benefit of wireless network coding [92] in terms of efficiency. The basic rationale of our approach is theoretically proved to be asymptotically optimal [85, 93], and experimentally shown to outperform other broadcast protocols with store-and-forward routing [92]. For sparse networks, we also propose dynamic adaptation, which prevents efficiency decrease for exceptional networks that are small and have low density [94].

Chapter 5

Real Life Implementation of Network Coding and Decoding Protocol

In this chapter, we present a practical protocol for efficient wireless broadcasting with network coding, which is derived from the proposed rate selection algorithms. In addition, we also propose a simple novel method for real-time decoding. As Codecast [15], CodeTorrent [16], MORE [17] and a file sharing protocol on Wireless Mesh Networks [18] present practical protocols using random linear coding, our protocol also uses random linear coding. While other protocols emphasize on loss recovery and delay constraint, our protocol focuses on energy-efficiency, which is achieved by our algorithms(IRON, IRMS and DRAGON). Our protocol also includes additional features: a termination protocol for eventual decoding at every node and a real-time decoding method.

5.1 Algorithm and Packet Format

In this section, we briefly describe our practical wireless broadcast protocol with a termination protocol. Algorithm 3 indicates basic operations of the protocol.

5.1.1 Framework for Broadcasting with Random Linear Coding

Basic operations of the protocol are described in the following Algorithm 3. As described in Algorithm 3, the source initiates broadcasting by sending its original data packets with a header specified in Table 5.1.3. When the source sends the data packet, it produces data of a prefixed size (size = κ) using padding if necessary. Other nodes initiate transmission of encoded data upon receiving the first coded packet, and stay in a transmission state where they will transmit packets with an interval decided by rate selection algorithms. Precisely, when intermediate nodes receive a data packet that is a source packet or a coded packet, they start scheduling encoded data transmission. The scheduling interval is decided by our algorithms (IRON, IRMS and DRAGON). Transmitted packets by intermediate nodes are coded packets generated using random linear coding with a header specified in Table 5.1.3 (See section 3.2.2 for a detailed description of random linear coding). Data transmission stops until nodes detect termination using a termination protocol.

Algorithm 3: Brief Description of data transmission Algorithm

3.1 **Source data transmission scheduling:** the source transmits sequentially the D vectors (packets) of a generation with rate C_s.

3.2 **Nodes' data transmission start conditions:** the nodes start transmitting a vector when they receive the first vector.

3.3 **Nodes' data storing condition :** the nodes store a received vector in their local buffer only if the received vector has new and different information from the vectors that the nodes already have.

3.4 **Nodes' data transmission stop conditions:** the nodes continue transmitting until themselves **and their current known neighbors in their local information base** have enough vectors to recover the D source packets.

3.5 **Nodes' data transmission scheduling:** the nodes retransmit linear combinations of the vectors in its local buffer after waiting for a delay computed from the rate selection.

3.6 **Nodes' data transmission restart conditions:** When a node receive a notification to indicate that one neighbor node requires more vectors to recover the D source packets and it has already stopped data transmission, the node re-enters in a transmission state.

5.1.2 Termination Protocol

A network coding protocol for broadcast requires a termination protocol in order to decide the appropriate time to stop transmissions of coded packets while the termination protocol confirms eventual decoding at every node. Our precise terminating condition is as follows: when a node (a source or an intermediate node) itself and all its known neighbors have sufficient data to recover all source packets, the transmission stops. This stop condition requires information about the status of neighbors including their ranks. Hence, each node manages a local information base to store one hop neighbor information, including their ranks.

Algorithm 4: Brief Description of Local Information Base Management Algorithm

```
4.1 Nodes' local info notify scheduling: The nodes start notifying their neighbors of
    their current rank and their lifetime when they start transmitting vectors. The
    notification can generally be piggybacked in data packets if the nodes transmit a
    vector within the lifetime interval.
4.2 Nodes' local info update scheduling: On receiving notification of a rank and
    lifetime, the receivers create or update their local information base by storing the
    sender's rank and lifetime. If the lifetime of the node information in the local
    information base expires, the information is removed.
```

In order to keep up-to-date information about neighbors, every entry in the local information base has a lifetime. If a node does not receive notification for update before the lifetime of an entry is expired, the entry is removed. Hence, every node needs to provide an update to its neighbors. In order to provide the update, each node notifies its current rank and a lifetime to its neighbors. The notification is usually piggybacked in an encoded data packet, but could be delivered in a control packet if a node does not have data to send during its lifetime. A precise algorithm to organize the local information base is described in Algorithm 4

The notification of rank has two functions : ACK and NACK. When a node has sufficient data to recover all source packets, the notification works as ACK, and when a node needs more data to recover all source packets, the notification works as NACK. When the notification works as NACK, a receiver of the NACK could have already stopped transmission. In other words, the receiver acquires a new neighbor that needs more data to recover all source packets. In this case, the receiver restarts transmission. The restarted transmission continues until the new neighbor notifies that it has enough data, or until the entry of the new neighbor is expired and therefore removed.

5.1.3 Packet Format

Algorithm 3 and Algorithm 4 use two kinds of packets: a data packet and a control packet.

Data Packet In each packet, protocol-ID marks a packet type, and for a data packet, it is NC-DATA (=1). Optional fields are for real-time decoding described section 5.2 and are not essential for basic operation in Algorithm 3 and Algorithm 4. The optional fields are used only when protocol-ID is NC-DATA-EXT(=3).

Field name	Size (byte)
Protocol-ID	1
Generation identifier	1
Sequence number	2
Source rate	2
Generation size	2
Rank of a node	2
Lifetime	2
Coefficient field size	2
Data size	2
Neighbor Number (optional)	2
Highest index (optional)	2
Lowest index (optional)	2
Sliding Encoding Window Size (optional)	2
Coefficients	Generation size x a Field size
Data	Predefined size (κ-11(Header Size))

Table 5.1: Data Packet Format

Each field in Table 5.1.3 has the following meaning:

- **Generation identifier** is a marker to let intermediate nodes encode packets only belonging to the same generation.
- **Sequence number** marks the sequence number of source packets, and is 0 for encoded data packets.
- **Source rate** advertises the source rate.
- **Generation size** marks the generation size.
- **Rank of a node** denotes the current amount of innovative data of the transmitter
- **Lifetime** denotes duration that the information in this packet (that is, the rank and the fact that transmitter is a neighbor of a node receiving this packet) is valid.
- **Coefficient field size** denotes the size of the fixed Galois Field for coefficients.

- **Data size** denotes the size of fixed data, κ.
- **Coefficients** are generally 0 for source packets except for the column of the sequence number. For example, when the sequence number is seven and the generation size is twenty, the coefficients for the source packet are [00000001000000000000]. The coefficients for encoded data packets are a global encoding vector that is a linear combination of randomly chosen local encoding vectors, and global encoding vectors of encoding data in the local buffer of a node (See section 3.2.2 for a detailed description of random linear coding).
- **Data** contains an original data for a source data packet with the predefined size κ, or encoded data for a intermediate node. Notice that the size of the encoded data is also κ.

Control Packet In each packet, protocol-ID marks a packet type, and for a control packet, it is NC-CTL (=2). Optimal fields are used to speed up a termination phase and are not essential for basic operations in Algorithm 3 and in Algorithm 4. The optional fields are used only when protocol-ID is NC-CTL-EXT(=4).

Field name	Size (byte)
Protocol-ID	1
Generation identifier	1
Source rate	2
Generation size	2
Rank of a node	2
Lifetime	2
Neighbor Number (optional)	2
Requested data number (optional)	2
Requested data sequence number (optional)	2 × requested data number

Table 5.2: Control Packet Format

Each field in Table 5.2has the following meaning:

- **Generation identifier** is a marker to let intermediate nodes encode packets only belonging to the same generation.
- **Source rate** advertises the source rate.
- **Generation size** marks the generation size.
- **Rank of a node** denotes the current rank of a transmitter
- **Lifetime** denotes duration that the information in this packet (that is, the rank and

the fact that transmitter is a neighbor of a node receiving this packet) is valid.

Local Information Base For stop and restart conditions, a node maintains its local information base that contains its known neighbors with lifetime as seen in Table 5.3. Whenever a node receives notification of rank and lifetime through a data packet or a control packet, the node updates its local information base. If a transmitter of the notification already exists in the local information base of a receiver, the receiver node only updates the lifetime of the entry for the transmitter. If an entry for the transmitter does not exist in the local information base, the receiver creates a new entry for the transmitter and stores the rank and lifetime of the received packet in the created entry. If the new entry is created after the receiver node stops transmission, the receiver node checks whether the receiver node needs to restart transmission by referring to the rank of the transmitter saved in the new entry; transmission restarts only when the rank of the transmitter is lower than the generation size.

Field name	Size (byte)
Neighbor Address	4 or 6
Generation identifier	1
Generation size	2
Rank of node	2
Lifetime	2

Table 5.3: Local Neighbor Table

5.1.4 Performance Evaluation using NS-2

We execute simulations using ns-2 version 2.28 with the parameters in Table 5.4.

Figure 5.1 represents the results for an average of 7 random graphs for $M = 20$ and of 3 random graphs for $M = 10$ with the minimum and maximum relative efficiency with error bars when a node with most neighbors is selected as a source: Figure 5.1(a) represents results when the average neighbor number is $M = 20$ and Figure 5.1(b) represents results when the average neighbor number is $M = 10$.

As described in section 4.3.2, the relative efficiency is $\frac{E_{cost}}{E_{optimal}}$ where the denominator $E_{optimal}$ is the theoretical optimal cost to broadcast one unit data to the whole network. The theoretical optimal cost is obtained by solving a linear optimization problem. (see details in section 4.3.2) Hence, relative efficiency of the theoretical optimal solution (opt(th) in

Parameter	Value(s)
Number of nodes	200
Transmission range	250m
Network field size	1400m x 1400m (M=10),2080m x 2080m (M=20)
Antenna	Omni-antenna
Propagation model	Two way ground
MAC	802.11
Data Packet Size	512 including headers
Generation size	500
Field $\mathbb{F}_p$	$p = 2$
Avg. Simulation time (sim tick) , M=20	637
Avg. Simulation time (sim tick), M=10	655

Table 5.4: Simulation Parameters of NS-2 for Efficiency Evaluation

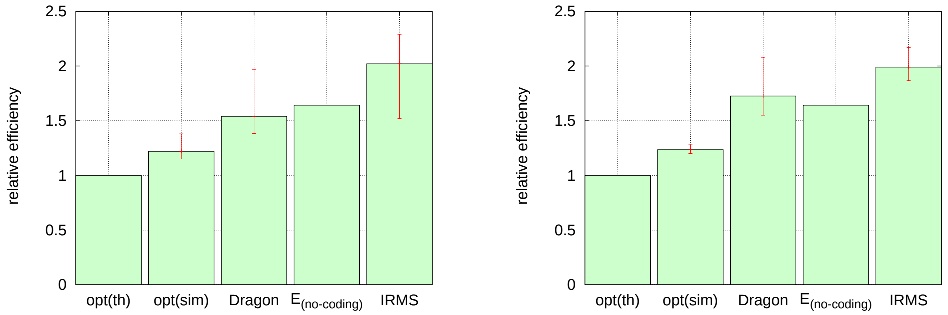
Figure 5.1) is always one. Opt(sim) represents a ratio of the theoretical optimal cost to the cost obtained by executing the simulations with the optimal solution. Because of the effect of the starting and ending phase and the effect of MAC and the physical layer, the opt(sim) is higher than one as seen in Figure 5.1.

As a reference, we use $E^{(\text{no}-\text{coding})}_{\text{rel}-\text{cost}}$ ($E_{(no-coding)}$ in Figure 5.1), which is the upper bound of efficiency without coding. We obtain it by reusing the argument of [73]. In a unit disk graph, two neighbors should be connected in order to forward packets and these two connected neighbors share a neighborhood area where any node inside the shared area receives duplicated packets as seen in Figure 4.5 in section 4.3.2. Considering the inefficiency of the shared area, $E^{(\text{no}-\text{coding})}_{\text{rel}-\text{cost}}$ is computed as a ratio between the maximally covered area with one transmission and the maximally covered area without duplication. (see section 4.3.2 for details), and the value is approximately 1.642.

One notable result is that our dynamic heuristic outperforms $E^{(\text{no}-\text{coding})}_{\text{rel}-\text{cost}}$ in some representative sparse networks and some representative dense networks.

5.2 Real-time Decoding

In this section, we propose a method for real-time decoding, which allows recovery of some source packets without requiring to decode all source packets at once. This section is organized as follows: we first explain the decoding process and the concept of real-time decoding in section 5.2.1, introduce our method for real-time decoding itself in section 5.2.2,



(a) Relative efficiency Avg. Neighbor Num (M)=20

(b) Relative efficiency Avg. Neighbor Num (M)=10

Figure 5.1: Relative efficiency of our heuristics

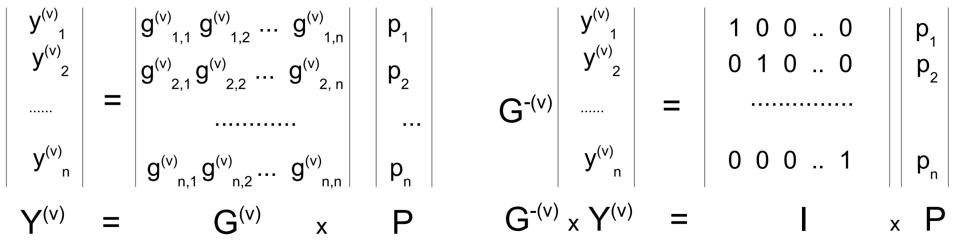
and then present experimental evaluation of the effect of our real-time decoding method SEW in section 5.2.3.

5.2.1 Overview of Real-time Decoding

In this section, we explain the general decoding process. Decoding is a process to recover the source packets from accumulated coded packets inside a node. As explained in section 3.2.1, any received coded packets are originated from the source, and are a set of linear combinations of the original source packets as represented in Figure 5.2(a). In Figure 5.2(a), $(p_j)_{j=1,\ldots,k}$ are k packets generated from the source, and the set $\{y_i^{(v)}\}$ is the set of packets that were received by a node v. The sequence of coefficients for $y_i^{(v)}$, $[g_{i,1}, g_{i,2}, \ldots, g_{i,k}]$ is the *global encoding vector* of the coded packet ($y_i^{(v)}$).

Considering the matrix of coefficients $[g_{i,j}]$ of a set of coded packets inside a node, the node can recover the source packets $[p_j]$ from the accumulated packets $[y_i^{(v)}]$ if the matrix of coefficients has full rank. Then, decoding amounts to inverting this matrix, for instance with Gaussian elimination as seen in Figure 5.2(b).

In the worst case, a node may have to wait until it has sufficient information to decode all packets at once (block decoding). Because block decoding delays recovery of source packets until the rank of a node reaches at least the generation size, the delay could be rather large. In order to shorten the delay of the block decoding, Chou et al. [54] suggested that an early decoding process could be possible by recovering some source packets before a node receives enough data for block decoding, but did not specify a method to ensure it. The early decoding process uses the fact that partial decoding is possible [54] if a subset



(a) A set of coded packets in a local buffer of a node

(b) Decoding with Gaussian Elimination with k=n

Figure 5.2: Decoding at a Node

of encoding vectors could be combined by Gaussian elimination, yielding a lower triangular part of the matrix as seen in Figure 5.3. Notice that packets forming the lower triangular part do not need to be on sequential rows inside the nodes' buffers and rows of the packet could be non-continuous in a matrix of the global encoding vectors and information vectors.

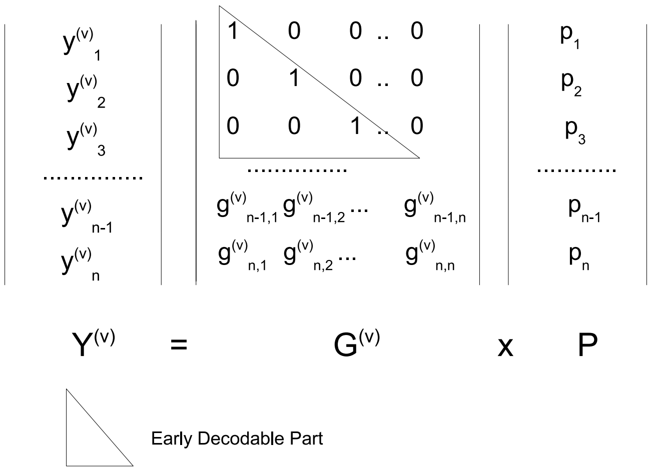


Figure 5.3: Low Triangle in Global Encoding Vectors in Local Buffer

An explicit mechanism to permit for early decoding is useful, since when the source rate approaches its "maximum broadcast rate", in other terms as the source rate approaches optimality, the probability of being able to partially decode after a fixed time decreases (as implied by [62]).

5.2.2 SEW (Sliding Encoding Window)

In this section, we introduce our real-time decoding method, Sliding Encoding Window (SEW). In order to enable real-time decoding, it ensures the existence of a low triangle in global encoding vectors saved in a node. Hence, the existence of an early decodable part as in Figure 5.3.

To ensure such existence, the method SEW relies on two properties:

Principles of SEW:

- SEW coding rule: generates only coded packets that are linear combinations of the first L source packets, where L is a quantity that increases with time.
- SEW decoding rule: when decoding, performs a Gaussian elimination, in such a way that one coded packet is only used to eliminate the source packet with the highest possible index (i.e. the latest source packet).

Before detailing the insights behind these rules, we first define notations: the high index of a node, I_{high}, and the low index of a node, I_{low}. As explained in section 5.2.1, a coded packet is a linear combination of source packets. If we assume that the most recently generated source packet always has the highest *sequence number*, that is if the source is successively sending packets p_1, p_2, p_3, ... with sequence numbers 1, 2, 3, ..., then it is meaningful to identify the highest and lowest such sequence number in the global encoding vector of any coded packet. Let us refer to the highest and lowest sequence numbers as: highest and lowest index of the coded packet respectively. For instance, for a packet $y = p_3 + p_5 + p_7 + p_8$, the highest index is 8 and the lowest index is 3.

Because all encoded packets have their own highest index and lowest index, we can also compute the maximum of the highest index of all not-yet decoded packets in a node, as well as the minimum of the lowest index. We refer to the maximum and the minimum as high index (I_{high}) of a node and low index (I_{low}) of a node. Notice that a node will generally have decoded the source packets from 1 up to its low index.

The intent of the SEW coding rule is to use knowledge about the state of neighbors of one node, namely their high and low index. A node restricts the generated packets to a subset of the packets of the source, until it is confirmed that perceived neighbors of the node are able to decode nearly all of them, up to a margin K. Notice that once all its neighbors may decode up to the first $n - k$ packets, it is unnecessary for the node to include packets $p_1, \dots p_k$ in its generated combinations.

Hence, the general idea of SEW is that it restricts the mixed original packets within an encoded packet from a window of a fixed size K. In other words, a node encodes only source packets inside a fixed Encoding Window as:

$$\text{i}^{th}\text{coded packet at node } v : y_i^{(v)} = \sum_{j=k}^{j=k+K} a_{i,j} p_j$$

where the $(p_j)_{j=k,\ldots,k+K}$ are K packets generated from the source. The sequence of coefficients for $y_i^{(v)}$ is the following global encoding vector:
$[0, 0, \ldots, a_{i,k}, a_{i,k+1}, \ldots, a_{i,k+K}, \ldots, 0, 0]$. A node will repeat transmissions of new random combinations within the same window, until its neighbors have progressed in the decoding process.

Our SEW is indirectly related to convolutional codes. In [95], Fragouli and Soljanin investigated connection between network coding and convolutional codes based on their subtree decomposition method. Our SEW does not directly use the state-space description of convolutional codes, but our SEW limits the number of coded symbols as an encoder of convolutional codes uses limited memory registers.

The intent of SEW's limiting the number of coded packets (symbols) is to guarantee proper functioning of the Gaussian elimination. An example of the SEW decoding rule is the following: assume that node v has received packets y_1 and y_2, for instance $y_1 = p_1 + p_9$ and $y_2 = p_1 + p_2 + p_3$. Then y_1 would be used to eliminate p_9 for newly incoming packets (the highest possible index is 9), and y_2 would be used to eliminate p_3 from further incoming packets. On the contrary, if the SEW decoding rule was not applied and if y_1 were used to eliminate p_1, then it would be used to eliminate it in y_2, and would result in the computation of $y_2 - y_1 = p_2 + p_3 - p_9$; this quantity now requires elimination of p_9, an higher index than the initial one in y_2. In contrast the SEW decoding rule guarantees the following invariant: during the Gaussian elimination process, the highest index of every currently non-decoded packet will always stay identical or decrease.

Provided that the neighbor state is properly exchanged and known, as a result, the combination of the SEW coding rule and the SEW decoding rule, guarantee that ultimately every node will be able to decode the packets in the window starting from its lowest index; that is, the combination guarantees early decoding.

Notice that improper knowledge of neighbor state might impact the performance of the method but not its correctness: if a previously unknown neighbor is detected (for instance due to mobility), the receiving node will properly adjust its sending window. Conversely, in our protocol, obsolete neighbor information, for instance about disappeared neighbors, will ultimately expire.

5.2.3 Real-Time Decoding: Effects of SEW

In order to evaluate the effects of our real-time decoding method SEW, simulations were run using an NS-2 simulator with parameters in Table 5.5 and the following additional parameters: SEW window size $K = 100$, high mobility (2.7 radio range/sec), and a source rate $M = 8.867$.

Parameter	Value(s)
Number of nodes	200
Transmission range	250m
Network field size	1100m x 1100m
Antenna	Omni-antenna
Propagation model	Two-way ground
MAC	802.11
Data Packet Size	512 including headers
Generation size	1000
Field $\mathbb{F}_p$, (xor)	$p = 2$

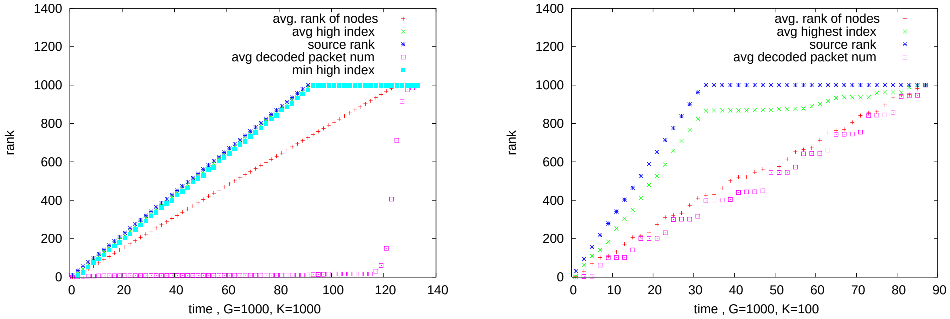
Table 5.5: Simulation Parameters of NS-2 for RDT Evaluation

We used two rate selection heuristics: IRON and DRAGON, and drew the evolution of the following parameters with time:

- Average rank of nodes,
- Average I_{high}
- Minimum I_{high},
- Source rank
- Average RTD : the average real-time decoding rate per unit time; the ratio between the number of decoded packets of a node and the rank of the node.

The results are represented in Figure 5.4(a) with rate selection IRON, whereas Figure 5.4(b) shows results using IRON and SEW.

As seen in Figure 5.4, SEW could decrease the gap between the average number of decoded packets and the average rank of nodes. Hence this evidences the success of real-time decoding: indeed, in that example, and thanks to this small gap, a node could decode more then 80 percent of received packets.



(a) IRON, without SEW

(b) IRON, with SEW

Figure 5.4: Evolution of avg. D, avg.(I_{high}),min.(I_{high}) and source rank, with high mobility, N=200 M=8.867

On the contrary, the results without SEW show that a node can decode only a fraction of its received coded packets for most of the simulation's duration (in the example, about 5 % for IRON, and 20 % for DRAGON), and will then decode most of the coded packets suddenly, at the end of the simulation. Such behavior is not uncommon: indeed the difference between being able to decode a whole set of packets or not may be made by one single additional packet.

In this spirit, we can explain the different decoding success rates by comparing the evolution of I_{high} and of the average rank of nodes. In the simulation without using SEW, the high index of a node I_{high} stays higher than the rank of the node and hence the node will not get a chance to perform real-time decoding: at the same time as the node gets more useful coded packets for the decoding process, it also gets additional coefficients to eliminate.

On the contrary, in the simulation using SEW, the average high index I_{high} increases more slowly than the rank of the source and at the similar pace with the average rank of nodes, as seen in Figure 5.4(b). This keeps I_{high} close to the rank. Therefore, in these simulations, nodes are able to decode more than 80 percent of received packets during almost the entire simulation time.

5.3 Summary

In this chapter, we describe a practical protocol specification derived from our algorithms (IRON, IRMS and DRAGON) detailed in chapter 4. The protocol description details algorithms to transmit encoded data packets and control packets with random linear coding as well as the packet formats and the local information base.

In our protocol, decoding at every node would be guaranteed with a termination protocol that requires a node and its known neighbors to have enough data to recover all source packets. This termination protocol needs precise known neighbors and the known neighbors could change because of the dynamic features of wireless networks. Hence, an algorithm for practical protocols must considers changing neighborhood for a clear termination protocol. For this, we proposed to maintain a local information base about one hop neighbors (precisely, their ranks) with an expiring lifetime for each neighbor entry. With the local information base, a node can distinguish current known neighbors from disappeared or completed neighbors, and clearly terminate encoded data transmission while confirming eventual decoding at every node.

In addition, we also proposed a real-time decoding scheme, Sliding Encoding Window (SEW) for a practical solution. A node is ultimately able to recover the source packets when the node receives sufficient data to recover all source packets at once. This ultimate decoding at the end of encoded data transmission in a network could cause intolerable prolonged delay. At worst, one lost packet can delay decoding of all received packets and the intolerable delay can make all received packets useless. Hence, it is important for a practical solution to support real-time decoding that recovers some portion of source packets before the end of encoded data transmission in a network. Our method using SEW supports real-time decoding by intentionally making an early decoding condition. Some simulation results of our simple method showed good real-time decoding rates. However, because there exist trade-offs between efficiency and real-time decoding rates, the real time decoding scheme should be combined with an appropriate rate selection method in order to obtain a high real-time decoding rate without losing much efficiency. This combination is one direction of our future work.

Chapter 6

Summary and Future work

This thesis studied efficient broadcasting for two kinds of traffic: control traffic and data traffic. The control traffic is specifically for mesh networks. Contents of the control traffic partially changes, and hence there exists redundancy between updating data and already updated date such as a database of mobile hosts (a so-called association database). On the other hand, the data traffic is a continuous flow of information such as video streaming, and is delivered in discrete packets containing discrete contents without redundancy. For these two kinds of traffic, we used two different approaches; for the control traffic we used a method based on classical store-and-forward routing and for the data traffic we used a new novel method, network coding.

For the control traffic, we extended efficient broadcasting with classical store-and-forward routing by taking into account redundancy between the updating data and the already updated data. More precisely, we utilized the benefit of a Multi-Point Relay (MPR) broadcast protocol, and proposed an association discovery protocol (ADP) that decreased the amount of broadcasting data, based on the fact that a set of stations (mesh clients) associated with one static node (a mesh router) slowly changed with time. With modeling and simulation we evaluated the performance of our protocol for the control traffic. The evaluation results showed that in some representative examples the consistency of broadcasted data could be maintained with decreasing overhead. This result could be generally applied to delay constrained traffic whose contents are already broadcasted and some particles of which are updated.

However, for continuous and discrete data traffic, there exists a new and novel approach for efficient broadcasting. It is network coding. Network coding may not be beneficial for the control traffic, but it could be beneficial for continuous and discrete data traffic. Hence, we explored efficient and simple broadcasting with network coding. As in Codecast [15], CodeTorrent [16], MORE [17], all-to-all wireless broadcasting [96] and a file sharing protocol on Wireless Mesh Networks [18], our exploration focused on a simple and practical solution that took advantages of random linear coding. Our practical solution especially focuses on efficient broadcasting and the efficiency is entirely decided by "a rate selection algorithm" that sets the average rate of each node. From the literature, it is known that the optimal rate selection can be obtained by solving a linear optimization problem formulated for a given network. However, the best-case-complexity-bound to solve the problem using Karmarkar's algorithm [66] is $O(N^7M^{3.5})$ where N is the total number of nodes and M is the expected number of neighbors. In addition, whenever topology changes because of mobility, newly changed optimization problems should be solved, and the average rate should be decided with exact consideration of packet losses. Moreover, there is no insight about the performance advantage of using network coding compared with other methods of using store-and-forward routing.

Instead of solving a linear optimization problem, we proposed a different and even simpler approach of using a rate selection algorithm, which was based on basic intuition and started from a hypothesis about the maximal-efficiency-benefit of wireless network coding: the efficiency-benefit of wireless network coding is maximized when every transmitted coded packet can bring new information to all receivers of the transmitted packet. Investigation of the conditions for achieving this goal showed us a tight coupling between a receiving rate (a max flow or a min-cut) of a node and the number of its neighbors. Inspired by the observation, we developed our approach that sets an identical rate (one) to other nodes except for the source rate (M) with the assumption that the number of neighbors is same as (M). The rate selection approach was denoted IRON.

The efficiency of our approach was theoretically studied in one dimensional torus networks and in two dimensional networks on Euclidean spaces. For the study of two dimensional networks, we specifically extended our basic approach (IRON) to increase rates of exceptional nodes (IREN/IRON) in order to handle some border effects that do not exist on torus networks. With IRON and IREN/IRON, we proved a maximum broadcast rate is asymptotically equal to the expected number of neighbors (M). The proof is deduced to the proof that the efficiency of our approach is asymptotically optimal. Moreover, from this result, we also deduced an important corollary: asymptotically, network coding could outperform other methods with store-and-forward routing. Theses results can be directly extended to any dimensional networks of Euclidean spaces.

These theoretical results are asymptotic and hence should be adapted for practical networks. The first adaptation is using local topology information. Precisely, the adaptation increases a rate for the most starving nodes where the most starving nodes are neighbors with the smallest neighborhoods. The static heuristic is denoted IRMS. In some experimental results, the efficiency of IRMS was, in practice, shown to outperform two other broadcast protocols with store-and-forward routing: an MPR-based broadcast protocol; and a CDS-based broadcast protocol with global topology and centralized algorithm to form a CDS set.

The static rate selection IRMS, generally showed good efficiency. However, we found limits of IRMS appearing in some networks with density irregularity. The limits occur because local topology information is not sufficient to find an appropriate rate selection for efficient broadcasting in strongly irregular networks. For such networks, an alternative approach has to be used. As the alternative solution we extended the static rate selection using local dynamic information, precisely the amount of innovative data (rank) inside each node. The dynamic heuristic is DRAGON: Dynamic Rate Adaptation from Gap with Other Nodes. DRAGON performs a feedback control, which intends to equalize the ranks of neighbors by increasing rates of nodes with more innovative data than their neighbors. DRAGON showed in some experimental results that it combats the problem that efficiency of our static heuristic decreases in sparse networks. Precisely, DRAGON, in some representative results, achieved 70 percent of theoretical optimal efficiency obtained by solving a linear optimization problem, and 85 percent of experimental optimal efficiency obtained by executing simulations with the optimal solutions.

Finally, we deduced a practical protocol for efficient broadcasting with network coding from these rate selection algorithms (IRON, IRMS and DRAGON), and presented a simple real-time decoding method that allows the decoding of some source packets before a node receives sufficient data to decode all source packets.

Overall, for continuous and discrete data traffic, this thesis proposed simple and practical solutions with network coding for efficient wireless broadcasting. Through theoretical studies, our basic approach was proved to be asymptotically optimal and to be at least as efficient as other broadcasting protocols with store-and-forward routing. Through experimental results, our static and dynamic heuristics based on the theoretical results were, in practice, shown to outperform the classical broadcast protocols using a decentralized algorithm, and to be at least as efficient as other classical broadcast protocols using a centralized algorithm. From these results, we can see that a simple rate selection algorithm enables efficient broadcasting with network coding, specifically for continuous and discrete data traffic.

Future prospects opened by this thesis include the study of broadcasting with network coding for continuous and delay-constrained data traffic. The study would focus on trade-offs between efficiency and delay to recover source data. Indeed, it is theoretically known that the possibility of recovering source data before the end of network coding transmission decreases as the efficiency of network coding increases. Hence, the trade-offs should be controlled depending on tolerable delay of data traffic. Further work also includes the study of the impact of mobility and transmission loss, and practical experiments of network coding using an actual testbed for real applications.

A file sharing application in wireless networks would be one of representative real applications that benefits from our network coding protocol in decreasing total time and energy to distribute data, as shown in simulation and modeling. If a file application executes on not all nodes but some nodes, these some nodes can organize a virtual network only including them and some extra nodes, and execute our protocol in the virtual network. In addition, when a wireless network is a wireless mesh network that has a hybrid structure and an infrastructure connection, we can support an efficient remote update system by using a node that connects a wireless multi-hop network and infrastructure as a source, and by disseminating a system file or software file through our protocol. Another application area is a car-to-car network. A car-to-car network would be a multi-hope wireless network connecting each car through wireless link and also have a infrastructure connection through nodes in some specific places such as a gas station or some electronic bulletin board. Heuristics of our protocol can be extended in order to use effectively this hybrid structure for efficient data dissemination. A car-to-car network also could have a node that could join in a group only for some duration before a car exits a road. In this case, if the node cannot recover encoded data received using network coding from the disjoining group, the received data could possibly be useless. This decoding-time constraint of encoded data using network coding can be supported by our real-time decoding method. In addition, because cars on the road could be irregularly distributed, a car-to-car network could be partitioned from time to time. In order to minimize this irregular disconnection effect, a protocol should dynamically adapt its operations. Moreover, our extension can use a special feature that cars running in both direction can bring data traffic to improve protocol efficiency. Our dynamic heuristic could be extended for the irregular connectivity and counter-direction traffic.

Bibliography

[1] Y.-S Chen Y.-C Tseng, S.-Y. Ni and J.-P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *In Proceedings of IEEE/ACM International Conference on Mobile Computing and Networking (MobiCom)*, September 1999.

[2] D. Cavin Y. Sassib and A. Schiper. Probabilistic broadcast for flooding in a wireless mobile ad hoc networks. In *In Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, March 2003.

[3] J. Y. Halpern Z. J. Hass and L. Li. Gossip-based ad hoc routing. In *In Proceedings of IEEE Computer and Communications Societies (INFOCOM)*, June 2002.

[4] S. Guha and S. Khuller. Approximation algorithm for connected dominating sets. In *In Proceedings of European Symposium on Algorithms*, pages 179–193, September 1996.

[5] P Jacquet C. Adjih and L. Viennot. Computing connected dominated sets with multipoint relays. *Ad Hoc & Sensor Wireless Networks*, 1(1):782–795, May 2005.

[6] T. Clausen, C. Adjih P. Jacquet(ed), P. Minet A.Laouiti, A. Qayyum P. Muhlethaler, and L. Viennot. Optimized link state routing protocol (olsr). *IETF RFC 3626*, October 2003.

[7] Shannon C. E. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, July 1948.

[8] S.-Y. R. Li R. Ahlswede, N. Cai and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[9] T. Ho, M. Medard R. Kotter, J. Shi M. Effros, and D. Karger. A random linear network coding approach to multicast. *IEEE Transactions on Information Theory*, 52:4413–4430, October 2006.

[10] S. Egner P. Sanders and L. Tolhuizen. Polynomial time algorithm for network information flow. In *In Proceedings of IEEE International Symposium on Information Theory (ISIT)*, 2003.

[11] T. Ho, D. Karger R. Koetter, M. Medard, and M. Effros. The benefits of coding over routing in a randomized setting. In *In Proceedings of IEEE International Symposium on Information Theory (ISIT)*, page 442, June 2003.

[12] P. A. Chou Y. Wu and S.-Y Kung. Minimum-energy multicast in mobile ad hoc networks using network coding. *IEEE Transactions on Communications*, 53(11):1906–1918, November 2005.

[13] D. S. Lun, M. Medard R. Rantnakar, D. R. Karger R. Koetter, E. Ahmed T. Ho, and F.Zhao. Minimum-cost multicast over coded packet networks. *IEEE/ACM Transactions on Networking*, 52(6):2608–2623, June 2006.

[14] C. Adjih S.Y. Cho and P. Jacquet. An association discovery protocol for hybrid wireless mesh networks. In *the 6th IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, June 2007.

[15] J.S. Park, D.S. Lun M. Gerla, and M. Medard Y. Yi. Codecast:a network-coding-based ad hoc multicast protocol. *Wireless Communication,IEEE*, pages 76–81, 2006.

[16] U. Lee, J. Yeh J.S.Park, and M. Gerla G. Pau. Codetorrent: Content distribution using network coding in vanets. In *In international Workshop on Decentralized Resource Sharing in Mobile Computing and Networking (MobiShare'06)*, 2006.

[17] S. Katti S. Chachuiski, M. Jennings and D. Katabi. Trading structure for randomness in wireless opporunistic routing. In *In Proceedings of annual meeting of the Special Interest Group on Data Communication (SIGCOMM)*, volume 19, pages 105–113, 2006.

[18] C. Barakat A.A. Hamra and T. Turletti. Network coding for wireless mesn networks: a case study. In *in International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '06)*, 2006.

[19] R. Koetter D. S. Lun, M. Medard and M. Effros. On coding for reliable communication over packet networks. In *Technical Report # 2741 MIT LIDS*, January 2007.

[20] B. Wiliams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *ACM Symposium on Mobile Ad Hoc Networking and Computing (MOHIHOC)*, June 2002.

[21] G. Tsudik C. Ho, K. Obraczka and K. Viswanath. Flooding for reliable mujlticast in multi-hop ad hoc networks. In *In proceedings of the international workshop on discrete algorithms and methods for mobile computing and communication (DIALM)*, pages 64–71, 1999.

[22] D. Maltz J. Jetcheva, Y. Hu and D. Johnson. A simple protocol for multicast and broadcast in mobile ad hoc networks, July 2001.

[23] H. Lim and C. Kim. Multicast tree contrusction and flooding in wireless ad hoc networks. In *In proceedings of the ACM international workshop on modeling, analysis and simultation of wireless and mobile systems (MSWIM)*, 2000.

[24] W. Peng and X. Lu. Efficient broadcast in mobile ad hoc networks using connected dominating sets. *Journal of software*, 1999.

[25] L. Lovasz. *On the ratio of optimal integral and fractional covers*. Discrete Mathematics, 1975.

[26] F. Dai and J. Wu. An extended localized alogirhtm for connected dominating set formation in ad hoc wireless networks. In *IEEE Transaction Parallel and Distributed Systems*, pages 908–920, October 2004.

[27] J. Wu and F. Dai. Broadcasting in ad hoc networks based on self-prunning. In *In Proceedings of IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[28] I. F. Akyildiz and X. W. Kiyon. Wireless mesh networks: A survey. *Computer Network Journal (Elsevier)*, pages 65–80, March 2005.

[29] Wireless lan medium access control (mac) and physical layer (phy) specifications, 2003.

[30] Terms and definitions for 802.11s, 2004.

[31] Functional requirements and scope for 802.11s, 2004.

[32] P. kinney. Ieee 802.15 general interest in mesh networking, November 2003.

[33] B.Lewis P.Piggin and P. Whitehead. Mesh networks in fixed broadband wireless access: multipoint enhancements for the 802.16 standard, July 2003.

[34] Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations, June 1999.

[35] E. Royer C.Perkins and S.Das. Ad hoc on-demand distance vector routing (aodv), July 2003.

[36] F. Templin R. Ogier and M. Lewis. Topology dissemination based on reverse-path forwarding (tbrpf), February 2004.

[37] D. Maltz D. Johnson and Yih-Chun Hu. The dynamic source routing protocol for mobile ad hoc networks (dsr), July 2004.

[38] Ieee 802.11 task group 3 draft 0.03 documents of the 802.11s working group., August 2006.

[39] X. Hong M. Gerla, G. Pei and T. Chen. Fisheye state routing protocol (fsr) for ad hoc networks, June 2001.

[40] M. Perman Z. J. Haas and P. Samar. The zone routing protocol (zrp) for ad hoc networks, July 2002.

[41] J. LI M. Jiang and Y. C. Tay. Cluster based routing protocol (cbrp), June 1999.

[42] Mario Gerla, Zhihan Lu Xiaoyan Hong, Kaixin Xu, and Charmaigne Flores. Lanmar + olsr: A scalable, group oriented extension of olsr. In *OLSR Interop and Workshop*, August 2004.

[43] J. Moy (ed.). Open shortest path first routing protocol (ospf) version 2, April 1998.

[44] J. Moy. Ospf anatomy of an internet routing protocol, January 1998.

[45] M.Nozaki H. Okada, K Mase and B. Zhang. Low-overhead and low-processing strategy of sta association information for ra-olsr in ieee802.11s. In *Transaction on The Institute of Electronics, Information and Communication Engineers (IEICE)*, March 2007.

[46] P. Jacquet T. Clausen and E. Baccelli. Ospf-style database exchange and reliable synchronization in the olsr. *INRIA Research Report RR-5283*, July 2004.

[47] Joseph Y. Halpern Zygmunt J. Haas and Li Li. Gossip-based ad hoc routing. In *In Proceedings of IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[48] Paul Muhlethaler Philippe Jacquet, Pascale Minet and Nicolas Rivierre. Increasing reliability in cable-free radio lans:low level forwarding in hiperlan. *Wireless Personal Communications*, 4(1):65–80, January 1997.

[49] J-Y. L. Boudec C. Fragouli and Jorg Widmer. Network coding : an instanace primer. 36:63–68, January 2006.

[50] R W Yeung S.-Y. R. Li and N. Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, February 2003.

[51] R. Koetter and M. Medard. Beyond routing:an algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, October 2003.

[52] S. Jaggi, P.Chou P. Sanders, K. Jain M.Effros, S.Egner, and L. Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Transaction on Information Theory*, 51(6):1972–1982, May 2005.

[53] P. A. Chou P. Sanders and K. Jain. Low complexity algebraic network multicast codes. In *In Proceedings of 15st ACM Symposium on Parallel Algorithms and Architecture*, 2003.

[54] Y. Wu P. A. Chou and K. Jain. Practical network coding. In *In Proceedings of 51st Allerton Conference on Communication, Control and Computing*, September 2003.

[55] A. Feinstein P. Elias and C. E. Shannon. Note on maximum flow through a network. *IRE transaction on Information Theory*, 2:117–119, 1956.

[56] L.R. Ford Jr and D. R. Fulkerson. Maximum flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[57] K. Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10:95–115, 1927.

[58] E. A. Dinic. Algorithm for solution of a problem of maximum flow in network with power estimation. *Soviet Math. Doklady*, 11:1277–1280, 1970.

[59] Jack Edmond and Richard M. Karp. Theoretical improvement in algorithm efficiency for network flow problems. *Journal of the ACM*, 19:248–264, 1972.

[60] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum flow problem. In *In Proceedings of ACM Symposium on Theory of Computing*, pages 136–146, 1986.

[61] C. Fragouli and E. Soljanin. *Network coding fundamentals.* Now, 2007.

[62] R. Koetter D. S. Lun, M. Medard and M. Effros. Further results on coding for reliable communication over packet networks. In *Internatoinal Symposium on Information THeory (ISIT)*, September 2005.

[63] C. Fragouli M. Jafarisiavoshani and S. Diggavi. On subspace properties for randomized network coding. In *Information Theory Workshop (ITW)*, July 2007.

[64] J. P Hubaux M. Canalij and C. Enz. The minimum-energy broadcast in all wireless networks: Np-completeness. In *In Proceedings of IEEE/ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 172–182, September 2002.

[65] J.S. Park, F. Soldo D.S. Lun, and M. Medard M. Gerla. Performance of network coding in ad hoc networks. In *In proceedings of Military Communication Conference (MILCOM) 2006*, pages 1–6, 2006.

[66] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, pages 373–395, 1984.

[67] D. jiang Z. Li, B. Li and L. C. Lau. On achieving optimal throughput with network coding. In *In Proceedings of IEEE Computer and Communications Societies (INFOCOM)*, pages 2184–2194, March 2005.

[68] S. Katti, H. Rahul D. Katabi, W. Hu, and M. Medard. The importance of being opportunistic: Practical network coding for wireless environments. In *In Proceedings of 43rd International Conference on Communication, Control and Computing*, 2005.

[69] S. Katti, W. Hu H. Rahul, M. Medard D. Katabi, and Jon Crowcroft. Xors in the air: Practical wireless network coding. In *In Proceedings of annual meeting of the Special Interest Group on Data Communication (SIGCOMM)*, 2006.

[70] S. Biswas and R. Morris. Exor:opportunistic multi-hop routing for wireless networks. In *In Proceedings of annual meeting of the Special Interest Group on Data Communication (SIGCOMM)*, 2005.

[71] J. Bicker D. S. J. De Couto, D. Aguayo and R. Morris. A high-throughput path metric for multi-hop routing. In *In Proceedings of IEEE/ACM International Conference on Mobile Computing and Networking (MobiCom)*, September 2003.

[72] C. Fragouli J. Widmer and J.-Y. L. Boudec. Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding. In *In Proceedings of IEEE Computer and Communications Societies (INFOCOM)*, 2005.

[73] J. Widmer C. Fragouli and J.-Y. L. Boudec. A network coding approach to energy efficient broadcasting: from theory to practice. In *In Proceedings of 1st intenational Workshop on Network Coding, Theory, and Applications (NetCod)*, 2006.

[74] Y. Li J. Wang and X. Wang. Network coding based multicast in internet. In *In Proceedings of IEEE/ACM International Conference on Parallel Processing Workshops*, page 44, 2007.

[75] C. Colbourn B. Clark and D. Johnson. Unit disk graphs. *Discrete Mathematics*, 86, December 1990.

[76] R. Gowaikar, B. Hassibi A. F. Dana, R. Palanki, and M. Effros. On the capacity of wireless erasure networks. In *In Proceedings of IEEE International Symposium on Information Theory (ISIT)*, page 401, June 2004.

[77] A.F. Dana, B. Hassibi R. Gowaikar, R. Palanki, and M. Effros. capacity of wireless erasure networks. *IEEE transactions on information theory*, 52:789–804, 2006.

[78] S. Deb, T. Ho M. Effros, R. Koetter D. R. Karger, M. Medard D. S. Lun, and N. Ratnakar. Network coding for wireless applications: A brief tutorial. In *In Proceedings of International Workshop on Wireless Ad-hoc Networks*, 2005.

[79] Y. Boykov and V. Kolmogorov. A experimental comparision of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 2004.

[80] Mosek, an optimization software for linear and convex optimization problems.

[81] Pushmeet Kohi and Philip H.S. Torr. Efficiently solving dynamic markov random fields using graph cuts. In *In Proceedings of International Conference on Computer Vision*, 2005.

[82] J. Shi A. Ramamoorthy and R. D. Wesel. On the capacity of network coding for random networks. *IEEE transactions on information theory*, 51:2878–2885, August 2005.

[83] J. Meng S.A Aly, V. Kapoor and A. Klappenecker. Bounds on the network coding capacity for wireless random networks. In *Third Workshop on Network Coding, Theory and Applications (Netcod)*, January 2007.

[84] R.A. Costa and J. Barros. Dual radio networks: Capacity and connectivity. In *Spatial Stochastic Models in Wireless Networks*, January 2007.

[85] S. Y. Cho C. Adjih and P. Jacquet. Near optimal broadcast with network coding in large sensor networks. In *In Proceedings of the first workshop on Information Theory for Sensor Networks*, June 2007.

[86] P. Barbe and M. Ledoux. *Probabilite.* Editions Espaces, 1998.

[87] M. Penrose. *Random Geometric Graphs.* Oxford Studies in Probability, 2003.

[88] M.S Kim I.K Lee and G. Elber. Polynomial/rational approximation of minkowski sum boundary curves. *Graphical Models and Image Processing*, 69:136–165, March 1998.

[89] R.J. Gardner and P. Gornchi. A brunn-minkowski inequality for the integer lattice. *Transcations of the American Mathematical Society*, 353:3995–4042, 2001.

[90] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, Inc., 1983.

[91] K. Voss. *Discrete Images, Objects and Functions in Z^n.* Springer-Verlag, 1993.

[92] C. Adjih S. Y. Cho and P. Jacquet. Heuristics for network coding in wireless networks. In *In Proceedings of International Wireless Networks (WICON)*, October 2007.

[93] S. Y. Cho C. Adjih and P. Jacquet. Near optimal broadcast with network coding in large homogeneous networks. In *INRIA Research Report RR-6188*, May 2007.

[94] S. Y. Cho and C. Adjih. Network coding for wireless broadcast: Rate selection with dynamic heuristics. In *INRIA Research Report RR-6349*, November 2007.

[95] C. Fragouli and E. Soljanin. A connection between network coding and convolutional codes. In *in IEEE International Conference on Communications*, 2004.

[96] E.F. Alaeddine, C. David S. Kave, and J.-Y. L. Boudec S. Yoav. A framework for network coding in challenged wireless network. In *in Demonstrations of International Conference on a Mobile Systems, Applications and Services (MobiSys '06)*, 2006.