2013-05-12

Cooperative Channel State Information Dissemination Schemes in Wireless Ad-hoc Networks

Wenmin He

Cooperative Channel State Information Dissemination Schemes in Wireless Ad-hoc Networks

by

Wenmin He

A Thesis
Submitted to the Faculty
of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science
in

Electrical and Computer Engineering
by

May 2013

APPROVED:

Professor Andrew G. Klein, Major Thesis Advisor

Professor D. Richard Brown III, Committee Member

Professor Xinming Huang, Committee Member

Abstract

This thesis considers a novel problem of obtaining global channel state information (CSI) at every node in an ad-hoc wireless network. A class of protocols for dissemination and estimation are developed which attempt to minimize the staleness of the estimates throughout the network. This thesis also provides an optimal protocol for CSI dissemination in networks with complete graph topology and a near optimal protocol in networks having incomplete graph topology. In networks with complete graph topology, the protocol for CSI dissemination is shown to have a resemblance to finding Eulerian tours in complete graphs. For networks having incomplete graph topology, a lower bound on maximum staleness is given and a near optimal algorithm based on finding minimum connected dominating sets and proper scheduling is described in this thesis.

Acknowledgements

I would like to express my gratitude to my advisor Professor Andrew Klein, who explored this uncharted territory with me. Your dedication and support were the pedestal for me to conduct research in a once strange field. The time we spent working together was one of the greatest part of my life, and I will forever benefit from what your insights, patience and dedication has done to me.

I also thank committee, Prof. D. Richard Brown III and Prof. Xinming Huang to serve as my committee member and examine my work.

Also, thank all my lab mates Ni Min, Raquel G. Machado, Wang Rui and Radu David. Thank you for all your help and support, and all the cheers you have passed on me.

I have to thank my family and all my friends in WPI, for all your care and love during my masters study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The issue of channel state information (CSI) — or knowledge of the channel gain and phase between transmitter and receiver — has been a long-standing concern in the design of wireless communications systems. The earliest communication systems, such as AM radio broadcast, employed non-coherent communication methods to avoid the need for CSI completely. To overcome the 3 dB penalty in using non-coherent schemes [1], communication systems subsequently transitioned to more sophisticated schemes employing coherent communication which required that the *receiver* have knowledge of the CSI, and this brought along the widespread use of coherent schemes such as pulse amplitude modulation (PAM) and quadrature amplitude modulation (QAM). As these schemes only require CSI at the receiver, a bulk of research activity flourished in the 1960's and 1970's in acquiring and estimating CSI at the receiver. By the late 1990's, spurred in part by interest in multi-antenna or MIMO technology (e.g. [2,3]) as well as bit-loading in multicarrier and OFDM systems [4], the research community recognized the gains possible by additionally exploiting CSI at the *transmitter*, and a wide variety of transmitter pre-coding schemes were developed which relied on the availability of channel state

information at the transmitter [5, 6]. The issue of CSI at both transmitter and receiver has been an active area of research, and people have begun to fully understand this issue in depth, including such aspects as how to acquire and track CSI, how many bits of feedback are required, the impact of delay, performance degradation due to imperfect CSI, and the overhead involved in maintaining accurate CSI at both the transmitter and receiver (see, for example, [7] and references therein).

As the research community shifted focus from purely point-to-point systems to networks of distributed nodes, the issue of accurate CSI availability became an even more prominent issue in the design of wireless communication networks. In the network setting, wireless nodes operate in an interference-limited regime, and consequently a wide variety of schemes are actively being studied to manage interference and achieve capacity, such as cooperative diversity [8, 9], coordinated multi-point (CoMP) [10], interference alignment [11, 12], relay networks [13], and distributed and coordinated beam-forming [14]. In the majority of these areas, the treatment of CSI generally follows a similar pattern. Initially, information theoretic results emerge, and often for analytical tractability these works assume that global CSI is available at all nodes in the network [8,9,11–13]. Subsequently, a host of suboptimal or near-optimal schemes are proposed which acknowledge the difficulty in attaining global CSI, particularly due to the amount of overhead, and they design around the problem of global CSI by considering less demanding schemes which require varying amounts of partial CSI throughout the network (for example, [15–21]). Another challenge that arises in distributed networks, particularly those where signals from multiple nodes are required to arrive in phase, is the need for distributed carrier synchronization [22].

As such, the convention wisdom is that having global CSI available at all nodes in a network is completely impractical, as evidenced by the large number of works which

seek to avoid this requirement. Indeed, in a network of N nodes where links between all pairs of nodes are assumed to be reciprocal, there are at most $(N^2 - N)/2$ such channels to estimate, disseminate, and track throughout all N nodes in the network. Clearly, since the number of parameters estimates throughout the network scales as N^3, this does seem to be a daunting task for large N, particularly in a mobile setting where the channel is likely to be time-varying. Nevertheless, there appears to be a gap in solidly understanding just how difficult it is to achieve global CSI throughout a network, even in cases where N is small.

At the same time, the need for interference mitigation has resulted in the consideration of networks of smaller amounts of nodes. In the cellular setting, femtocells have been a recent trend [23]. In the ad-hoc setting, hierarchical networks consisting of clusters of smaller amounts of nodes have shown promise [24]. Again, however, in these domains global CSI is generally either assumed for simplicity, or avoided due to the standard belief that the overhead is too onerous.

Nevertheless, a handful of works have considered the issue of disseminating global CSI throughout a network of nodes. In [25] distributed channel estimation is considered in the case of a sensor network with static channels. In that work, it is shown that knowledge of path-loss exponents can be exploited to give prior information about node locations, thereby reducing the order N^2 channel gains to one of order N. Using a combination of random sleep-cycling and expectation propagation, network energy consumption is minimized while providing accurate channel state information throughout the network. In [26], a multiuser relaying scheme is considered, and the overhead required for dissemination is analyzed for the case when relays are grouped into clusters. In addition [26] considers the case of static channels, and assumes that perfect channel estimation has already taken place. Note that the fields of *data dissemination* [27] and *gossiping* [28] have some connection

to the issue of disseminating CSI throughout a network; however, the majority of those works operate at higher network layers, and are not concerned with the issue of *estimating* and disseminating CSI. In summary, it appears that we lack a fundamental understanding of how to disseminate reliable CSI throughout a network consisting of *time-varying* channels, and in quantifying exactly how much overhead is involved.

This thesis will address the issue of global CSI in wireless networks, and will answer two fundamental questions

- How would one jointly estimate and disseminate CSI throughout a network, including phase information, and what are the best strategies?
- Given a performance metric in evaluating CSI dissemination and estimation schemes in a network with known topology, what are the bounds of such a metric? Are these bounds achievable?

A more solid understanding of these fundamental questions will impact the design of future wireless networks, particularly those using small number of distributed cooperating nodes. To obtain preliminary results of a possibly proliferated yet immature field of study, this thesis uses simplified models of channel and exchanged messages. This thesis also makes ideal assumptions of perfect estimation and perfect dissemination processes to deduce elegant and mathematical tractable results.

Chapter 2

Background

This chapter provides general background of assumptions and settings related to solving CSI estimation and dissemination problem. The system model and a feasible channel model is discussed in the first section, followed by an example of a trivial solution for a 3-node wireless network. Necessary backgrounds of graph theory is also given in this chapter.

2.1 System Model

This paper studies a system model consisting of a cluster of N wireless nodes communicating over a time-varying flat-fading channel with additive white Gaussian noise (AWGN). Assuming the channels between all nodes are reciprocal, the network consists of L complex channel gains ($L \leq (N^2 - N)/2$) between all pairs of nodes. Because of the potential communication schemes these nodes intend to use due to numerous applications, each of the N nodes in the network requires the global channel state information, which is the knowledge of L complex channel gains. Estimating and disseminating these channel gains in a efficient way so that each node gets a good knowledge of global channel state information is of interest here.

Consider the channel model of node j receives the symbol $x_i[n]$ from node i through the complex channel gain $h_{i,j}[n]$ at time n. The received signal $y_j[n]$ is observed:

$$y_j[n] = h_{i,j}[n]x_i[n] + w_j[n]$$

where $w_j[n]$ is AWGN, $h_{i,j}[n]$ is the complex channel gain at time n for some $i, j \in \{1, ...N\}$ and $i \neq j$. The channel between each pair of nodes is assumed reciprocal so that $h_{i,j}[n] = h_{j,i}[n]$. Under such an assumption there are L complex channel gains to be estimated and each node maintains its own estimate of the state of all channels. Denote the kth node's estimation of channel gain between (i, j) as $\hat{h}_{i,j}^{(k)}$. Another assumption is that if there exist a channel gain $\hat{h}_{i,j}$, the received SNR is above some threshold so that the maximum reliable transmission rate supported by the channel R equals the channel capacity [29], such that for any encoded data below this rate, there exists a transmission scheme so that the data can be decoded with arbitrarily low probability of error [29]. Under such assumption, the dissemination through the link i, j will always be considered successful. This assumption will simplify the complexity of the system model, leading to mathematical tractable solutions. For similar concerns, the estimation made by each node from decoding the training sequence will always be considered perfect, or at least the estimation errors are not concerns of this thesis.

Due to the assumption of the network, a single transmission at time n from node i serves the following two purposes:

- **Estimation:** Node j ($j \neq i$) receives transmission from node i if there exist a channel gain $\hat{h}_{i,j}$, and the true channel gain $h_{i,j}[n]$ can be *instantaneously* observed by node j to form its estimate $\hat{h}_{i,j}^{(j)}[n]$, it is called instantaneous estimate. Notice that only $\hat{h}_{i,j}^{(i)}$ and $\hat{h}_{i,j}^{(j)}$ can be instantaneous estimates.

- **Dissemination:** The transmission from node i will include the coded estimate of *one* of the L channel gains, for example, $\hat{h}_{i,j}^{(i)}[m], m < n$. Node k ($k \neq i$) receives and decodes the disseminated estimate from node i and uses this to form its own estimate $\hat{h}_{i,j}^{(k)}[m]$.

2.1.1 Data Transmission Mode

Notice that the above assumptions only allow one node transmit one of the L estimates of channel state in a single time step. Although there exists techniques which enables multiple nodes communicate at the same time, restrictions are nevertheless put on the dissemination considering potential energy constraints and the impact on the transmissions of data payload. Additionally, it is assumed that the system splits transmission into two epochs as shown in Fig. 2.1: one where the estimation, dissemination take place, and one where data is actually transmitted. The epochs for estimation and dissemination contain blocks of symbols used by training sequence and encoded CSI, and should be short enough so that the channel gain h can be treated as time-invariant during these epochs. As the focus is on fundamental problem of estimation and dissemination, no restrictions are played on the data transmission mode other than the fact that it needs to be periodically interrupted to permit continued channel tracking and dissemination.

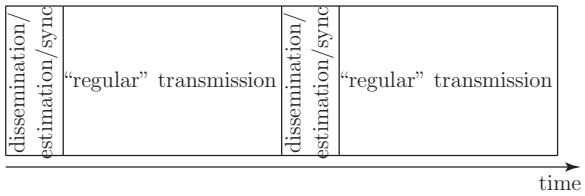


Figure 2.1: Alternating epochs of dissemination/estimation/sync

2.1.2 CSI Dissemination Protocol

Now proceed to consider how dissemination and estimation might operate for this simplistic 3-node setup. In the first round, assume there is no knowledge of CSI anywhere in the network. At time $n = 0$, let s_1 transmits known training data. In this case, s_2 will then form an estimate $\hat{h}^{(2)}_{1,2}[0]$ of the true channel $h_{1,2}[0]$, while s_3 will form an estimate $\hat{h}^{(3)}_{1,3}[0]$ of the true channel $h_{1,3}[0]$. In the second time slot

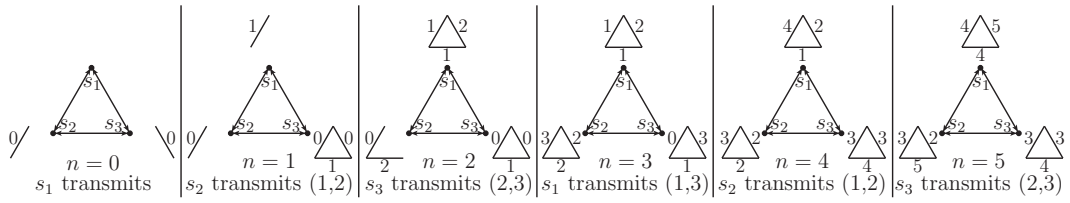


Figure 2.2: Operation of protocol for 3 node case. Numbers on edges indicate time of most recent information used in estimating each gain locally at each node.

when $n = 1$, the time-varying channels may have changed somewhat, depending on the operating environment. Nevertheless, node 2 could proceed by transmitting its estimate of the $(1, 2)$ channel $\hat{h}^{(2)}_{1,2}[0]$ along with training data. Node 1 and 3 decode the data and form their own estimates $\hat{h}^{(1)}_{1,2}[0]$ and $\hat{h}^{(3)}_{1,2}[0]$ using stale estimate $\hat{h}^{(2)}_{1,2}[0]$. In addition, node 1 and 3 instantaneously estimate the true channel gain $h_{1,2}$ and $h_{1,3}$ respectively from the training data sent by node 2. As much, node 1 has two pieces of information that can be used to form the estimate $\hat{h}^{(1)}_{1,2}[1]$. By the end of the second time slot, 1 has $\hat{h}^{(1)}_{1,2}[1]$, 2 has $\hat{h}^{(2)}_{1,2}[0]$ and 3 has $\hat{h}^{(3)}_{1,2}[0]$,$\hat{h}^{(3)}_{1,3}[0]$,$\hat{h}^{(3)}_{2,3}[1]$.

Next, in the third round at time $n = 2$, node 3 transmits its now stale estimate of the $(2, 3)$ link, giving node 1 a stale estimate of the $(2, 3)$ link and a instantaneous estimate of the $(1, 3)$ link.

Meanwhile node 2 obtains both a stale and a current estimate of the $(2, 3)$ link due to the dissemination and local estimation, respectively. Finally, the nodes continue repeating this same sequence of steps which are summarized in the 3-round

protocol shown in figure 2.2.

Naturally, a **protocol** is defined as a sequence of dissemination/ estimation rounds as discussed in previous example, where an periodic 3-round protocol was shown in Table 2.1.

Table 2.1: 3-node protocol

transmitting node	disseminated channel
1	(1,3)
2	(1,2)
3	(2,3)

Notice that due to the assumption of dissemination and estimation, there is no way that every node i will always keep the instantaneous, or the up-to-date estimates for all L complex channel gains. At time n, if the node k has a local estimate $\hat{h}_{i,j}^{(k)}[m]$, $i \neq j$ based on the observations made up through time m where $m < n$, this estimate is called **stale**[1]. Define the **staleness** of an estimate as the difference in the current time and the time of the most recent observation used in forming the estimate for a given link. For example in Fig. 2.2, the staleness of node s_2's estimate of the $(1, 2)$ link at time $n = 5$ is $5 - 3 = 2$. In general, an estimator will not only use the most recent information in forming an estimate, but also previous information to exploit the correlation of time-varying channel statistics. This is particularly true in recursive estimators such as the Kalman filter. Nevertheless, the staleness is adopted as a proxy to evaluate the estimate performance and this thesis tries to find ways to design protocols minimizing the worst case staleness as well as average staleness. After 4 rounds, i.e. for $n \geq 3$, the protocol reaches a steady state where all nodes have an estimate of all links, and those estimates are based on information with a **maximum staleness** of 3 time slots. In addition, the **average staleness** across all links and nodes also reaches a steady state value of $4/3$ for $n \geq 3$. This

[1]Though predictions can be done in multiple ways, it is not our concern

protocol in Table 2.1 is claimed to be an optimal protocol for the 3-node case in terms of minimizing maximum staleness *and* minimizing average staleness.

Disseminating the CSI of one of the L links in a given time is performed as broadcasting in the network. Constructing a CSI dissemination protocol is somehow different from constructing a broadcasting routing scheme. Since every nodes needs constant update of the global knowledge of CSI, the dissemination process is also expected to be constantly executed and initiated by every node in the network. Also, given the presence of estimation, the message needs to broadcast is dynamically alternating based on the previous CSI disseminated and the path chosen for broadcasting. Meanwhile, since the goal of CSI dissemination protocol is to treat every piece of CSI unbiased, and obtain the fresh estimate global channel states at all times, a CSI dissemination protocol is expected to be periodic given a fixed network topology to satisfy above conditions.

A number of literature [30–32] focused on routing protocol designs for Ad-hoc wireless networks, especially routing using connected dominating set. Though similarities of these two problems can be found, CSI dissemination protocol is demanded to provide optimal broadcasting path for the next piece of information after completing the previous broadcast and hopefully, without any additional phase of estimation through sending training sequences, which is not a requirement in generic routing problems. This could be a dynamic optimization problem in terms of routing path chosen and the network topology. In the rest of the thesis, the solution is provided by imposing additional requirements in forming connected dominating sets and scheduling the routing path using these connected dominating sets.

2.2 Basics of Graph Theory

As the CSI estimation and dissemination problem stays open as it is, the thesis nevertheless provides partial solutions based on principles of graph theory. In this section, basic notions and related topics of graph theory is briefly introduced.

In this thesis, an undirected graph G, consists of vertex set $V(G)$ and edge set $E(G)$, is used to model the topology of an arbitrary wireless networks being studied, based on the reciprocal links assumption. The size of vertex set $V(G)$ is N, corresponding to the number of nodes in a wireless network while the size of edge set $E(G)$ is L, corresponding to the number of links in a wireless network. The size of vertex set $V(G)$ is also called the order of G.

The neighboring neighborhood (or open neighborhood) of a vertex $v \in V(G)$, denoted by $N(v)$, is the set of vertices adjacent to v:

$$N(v) = \{x \in V | vx \in E\}.$$

The degree of v, denoted by $deg(v)$, is the number of edges incident with v.

2.2.1 Complete Graph

A complete graph of order N is denoted as K_N. For every $v, x \in V(K_N)$, there exists $\{vx\} \in E(G)$. Examples of complete graphs are shown below in Fig 2.3.

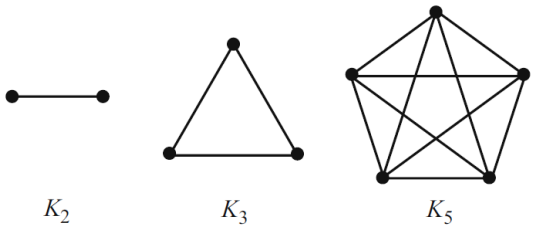


Figure 2.3: Examples of Complete Graphs

2.2.2 Spanning Tree

A graph is connected if every pair of vertices can be joined by a path. A tree is a connected graph containing no cycles. A tree of order N contains $N-1$ edges. Nodes with degree of 1 are called leaves in a tree. Any tree with order $N \geq 2$ has at least 2 leaves. A tree is called a rooted tree if one vertex has been labeled as root, in which case the edges have a natural orientation, towards or away from the root. A rooted tree example is given in Fig. 2.4.

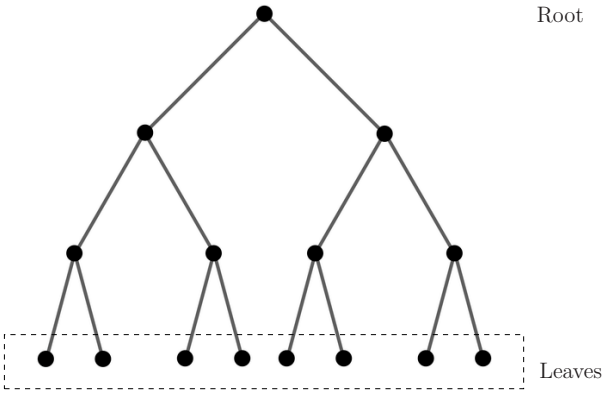
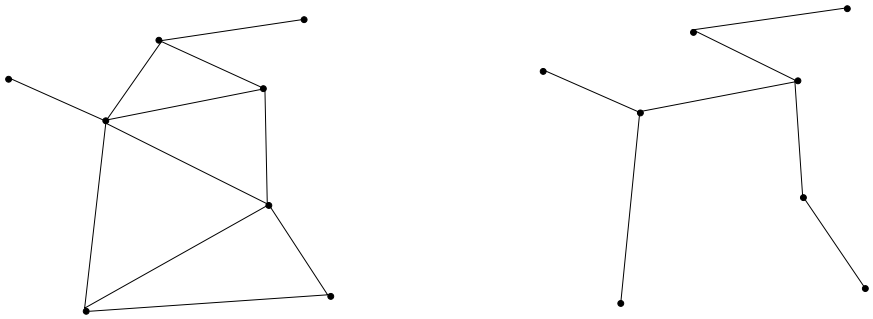


Figure 2.4: A rooted tree

Given a graph G and a subgraph T, T is a spanning tree of G if T is a tree and contains every vertex of G. A spanning tree can find its use in logistics, network routing, traffic etc. A example of a graph and its spanning tree is shown in Fig. 2.5. Notice that the spanning tree of a graph is usually not unique.

2.2.3 Dominating Set

Definition 1. *Let $D \in V(G)$, if $\forall u \in V(G)$, either $u \in D$ or $u \in N(D)$, then D is called a dominating set of G. If any proper subset of a dominating set is no longer a dominating set, then the dominating set is called a minimum dominating set. The dominating sets of G containing the least number of vertex is called the*



Original Graph Spanning Tree

Figure 2.5: A graph and its spanning tree

least dominating set.

For example, in Fig. 2.6, $D_0 = \{v_0\}$, $D_1 = \{v_1, v_4, v_6\}$, $D_0 = \{v_1, v_3, v_5, v_6\}$ are all dominating sets of G. The first two are minimum dominating set, while D_0 is the least dominating set.

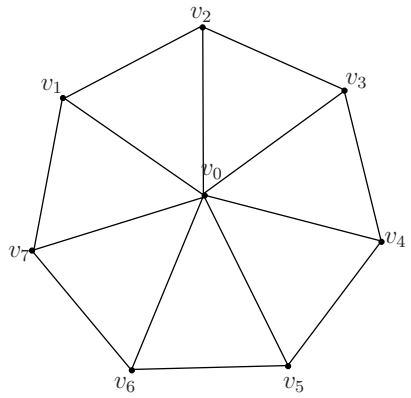


Figure 2.6: A graph and its dominating sets

Chapter 3

Minimum Staleness Protocol Design in Networks with Complete Graph Topology

First examine the solutions of CSI dissemination problem in complete graphs. This thesis ought to find a way to design protocols which minimizes staleness resulted from such transmission sequence. A wrongfully designed protocol may result in the staleness of some of the estimates to be unbounded, that is some of the estimates may never updated on particular links. Below is the conditions of a valid protocol.

A valid protocol is defined to be a transmission sequence that results in a **bounded** worst case staleness for a network. A valid protocols should satisfy the following properties:

1. Each of the L channel information must be disseminated by some node at least once in a finite period, to ensure that the kth node forms estimates of $h_{i,j}$ where $i \neq j \neq k$.

2. Each node must transmit at least once in a finite period. If, for example, node

i never transmitted, then any other node j would not be able to estimate (or receive through dissemination) $h_{i,j}$.

3. Any node i should only ever disseminate estimates $\hat{h}_{i,j}^{(i)}$. Since the staleness of $\hat{h}_{i,j}^{(k)}$ must be greater or equal to $\hat{h}_{i,j}^{(i)}$ and $\hat{h}_{i,j}^{(j)}$.

3.1 Worst Case Staleness

3.1.1 Properties of Minimum Staleness Protocols

Here we try to find transmission sequences that minimize worst case staleness, despite the fact that there are other potential metrics which can be considered as proxy for performance evaluation. First examine the properties of such protocols:

Theorem 1 (Minimum Worst Case Staleness Protocols are L-Periodic)**.** *The minimum worst case staleness for an N nodes complete network cannot be less than L. The protocol which achieves this bound must be periodic, and the period is also L.*

Proof. It is trivial to claim that estimates $\hat{h}_{i,j}^{(k)}$ will have staleness greater or equal to $\hat{h}_{i,j}^{(i)}$ since the latter can be updated through direct estimation while $\hat{h}_{i,j}^{(k)}$ can only be updated through receiving disseminated estimates $\hat{h}_{i,j}$. Since in our settings, only one estimate can be disseminated in a single time slot, the minimum period of transmission to cover all L links is L. The length of the period determines that the minimum worst case staleness can be no less than L. A protocol achieving this bound must always disseminate the freshest estimates, and also L-periodic since $\hat{h}_{i,j}^{(i)}$ and $\hat{h}_{i,j}^{(j)}$ cannot both be freshest estimates at the same time. □

Based on the conditions of valid protocols, the dissemination process can be defined as follow:

Definition 2. *In a N-node network, the process of node i transmitting the estimate concerning $h_{i,j}$ is represented as visiting the edge $(j,i) \in E$, where E is the edge set of G and $G(V,E) = K_N$ (K_N is the complete graph with N vertices). A valid protocol is a traverse of G.*

The motive of using the graph representation is spurred from the geometric nature of this problem. Since a valid protocol will restrict node i to transmit estimates $\hat{h}_{i,j}$ only, it is suffice to describe the dissemination of estimate $\hat{h}_{i,j}$ by node i as visiting the edge (j,i). Since the network is complete, dissemination of any arbitrary link will be received by every other node to form their own estimates. Thus a traverse of the graph G is equivalent to forming a valid protocol.

3.1.2 Minimum Staleness Protocol Design

Consider a protocol achieving the lower bound of worst case staleness is represented as an edge sequence $\{e_1, e_2, ..e_L\}$, $e_i \in E(G)$. The condition that such a protocol will always use the freshest estimates for transmission requires the edges to be adjacent. Nevertheless, since the protocol is a period L sequence containing every $e_i \in E(G)$, it can be treated as an Eulerian cycle in graph G. So finding Eulerian cycles on the complete graph K_n is equal to finding protocols achieving lower bound of worst case staleness for an N-node network.

According to *theorem 1.20* in [33], a complete graph K_{2k+1} with even degree on every vertex contains at least one Eulerian path. In order to find these paths, one can apply the classic Hierholzer's algorithm or Fleury algorithm [33] to find Eulerian cycles. However K_{2n} is not Eulerian [33], which means the protocol achieving the lower bound of worst case staleness L does not exist. So for N-even cases, we loose our constraint to allow transmission of estimates which are 2 time slots stale (freshest estimates are 1 time slot stale).

Allowing transmission of estimates that are not most recently formed will perform as branching from the vertex passed two time slots ago and leave the last traversed edge to be a dead end edge. An example is shown below as Fig. 3.1. A protocol designed under the above assumptions is equivalent to decompose K_N into an Eulerian sub-graph and dead end edges, while trying to minimize the number of dead end edges. One way to achieve this goal is building a protocol based on decomposing K_{2n} into n disjoint dead end edges and a Eulerian sub-graph consisting of $n-1$ Hamiltonian cycles. [34] The protocol will result in the worst case staleness to be $L+1$ for $N/2L$ of the time, and L for the rest.

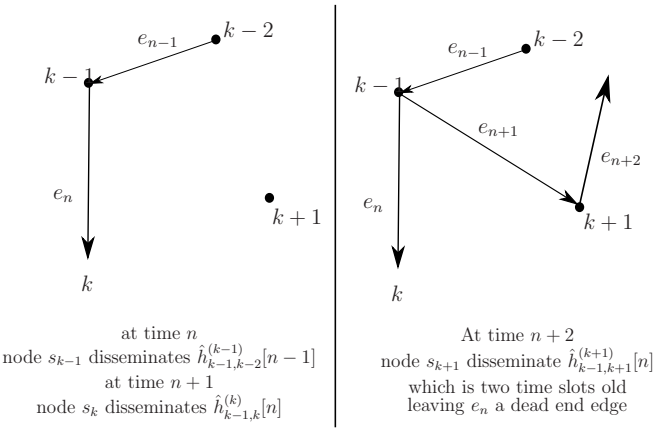


Figure 3.1: Finding minimum worst case staleness protocols for N-even networks. Alowing transmission of estimates that are two time slots old will result in dead end edges, forming a Eulerian subgraph

Above is the guideline for designing protocols minimizing worst case staleness in a complete graph. Designing such protocols relies on finding Eulerian tours which can be done in multiple ways. The following chapter describes a formulated approach designing protocols not only minimizes worst case staleness, but also average staleness.

3.2 Average Staleness

To evaluate the overall performance of the estimates under a protocol minimizing the worst case staleness, the **average staleness** of a protocol is examined.

3.2.1 Average Staleness Lower Bounds

Lemma 2 (Lower bound of the average staleness when N is odd)**.** *The average staleness of a N node network is greater than or equal to $S^* = (N^3 - 3N^2 + 8N - 8)/4N$, when N is odd*

Proof. For each edge i, j, there are $N-2$ estimates, namely $\hat{h}_{i,j}^{(k)}$, only update when receiving disseminated estimates thus they are called indirect estimates. Since we only allow disseminating one of these estimates, the staleness of $\hat{h}_{i,j}^{(k)}$ are different under different pairs of i, j. The case when these estimates have the minimum sum of staleness is the staleness of these estimates take distinct values from 1 to L for different pairs of i, j. There are also $N-1$ direct estimates $\hat{h}_{i,j}^{(j)}$ given an i, and the staleness of $\hat{h}_{i,j}^{(j)}$ are different under different i since we only allow one node transmitting in any given time slot (whatever node i transmits, all other $N-1$ nodes can instantaneously estimates the link between them and i). The case when these estimates have the minimum sum of staleness is the staleness of these estimates take distinct values from 0 to $N-1$ under different j. Computing the average staleness in the best case yields $S^* = [(1+L)L(N-2)/2 + (0+N-1)N(N-1)/2]/(NL)$, where $L = N(N-1)/2$, which yields the same result described in the lemma. □

Denote $S^* = (N^3 - 3N^2 + 8N - 8)/4N$ as the lower bound of average staleness. It is easy to construct the lower bound of average staleness for K_N when N is even.

Corollary 3 (Lower bound of average staleness when N is even)**.** *The average*

staleness of a N node network is greater than or equal to $S^* + (N-2)/N(N-1)$*, when N is even.*

Proof. In each time slot, there are $(N-2) * N/2$ estimates updated by receiving estimates which had staleness of 2 by the time they were used for transmission. This procedure gives an additional average staleness of $(N-2)/N(N-1)$, compared to the case when only the freshest estimates are used for transmission. □

3.2.2 Formulated Protocol Design

An arbitrary protocol found by applying Hierholzer's algorithm or Fleury's algorithm can result in unbounded average staleness. Thus we found an alternative way to design minimum worst case staleness protocols based on Hamiltonian decomposition, by which the maximum average staleness is less than $S^* + 0.5$.

According to Walecki [35], a complete graph K_n can be decomposed into:

- m Hamiltonian cycles, or m Hamiltonian paths and an almost-one factor. When $n = 2m + 1$
- m Hamiltonian paths, or $m - 1$ Hamiltonian cycles and a one-factor. When $n = 2m$

A "zigzag" approach can be used to design optimal protocols for $N = 2m + 1$. Let s_{i_n} denote the node that transmits at time n. In the first time slot $n = 1$, the node s_1 disseminates $\hat{h}^{(1)}_{1,N}$, which is its estimate of the $(1, N)$ link so that $i_0 = N$ and $i_1 = 1$. In every subsequent time slot, node i_n disseminates its estimate of the

(i_n, i_{n-1}) link and

$$\begin{cases} \{i_1, ..i_{N-1}\} = \{1, N-1, 2, N-2, .., (N+1)/2\} \\ i_{mN+k} = [(i_k + m - 1) \bmod (N-1)] + 1 \\ i_{mN} = N \end{cases}$$

Where $1 \leq m \leq (N-3)/2$ and $1 \leq k \leq N-1$. This approach was also used by Wegman in 1990 [36].

Lemma 4 (Upper bound of average staleness)**.** *The above protocol construction for* $N = 2m+1$ *results in the upper bound of average staleness* $S^* + (N^2-1)/2N^2$

Proof. Since the protocol is L-periodic and disseminates estimates concerning L links once every period, estimates $\hat{h}_{i,j}^{(k)}$ have different staleness ranging from 1 to L for different pairs of i, j at any given time. The staleness of estimates $\hat{h}_{i,j}^{(i)}$ differs for different j and depend on the interval between the current time slot and the time slot when node j doing the most recent transmission, and we let these intervals to be $t = t_{s1}, ...t_{sN}$. Consider two adjacent length N sequence S_1 and S_2 defining the node number doing the transmission, each of the two sequences contains the set $[N]$ as shown in Fig.3.2. Let the interval between current time slot and the tail of S_1 to be x, which contains x distinct elements forming the set $[X]$. Let the subsequent x elements to form the same set $[X]$ so that the sum of t can reach the maximum $x(x-1)/2 + (2x+N+x-1)(N-x)/2$. Take the derivative and let it be zeros yields $x = N/2$, thus the average staleness upper bound is attained. □

For cases when $N = 4k+2, k > 0$, a similar algorithm for designing protocols based on Hamiltonian path decomposition is shown below. First define two sequences $i_1, i_2.., i_L$ and $j_1, j_2, .., j_L$. Let $[i_n, j_n]$ denote node i_n disseminates its esti-

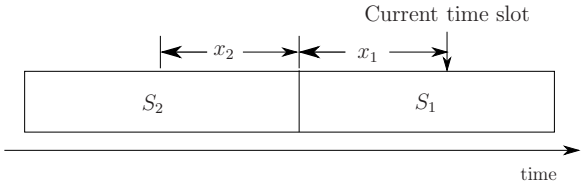


Figure 3.2: S_1 and S_2 are the sequence of node numbers doing the dissemination and each contains every N node. Assuming current time slot lies on S_1, only when x_1 and x_2 spans the same set $[X]$ could the sum of the interval between each N node and current time slot reaches maximum

mate of the (i_n, j_n) link

$$\begin{cases} \{i_1, ...i_{N-1}\} = \{1, N, 2, N-1, .., N/2\} \\ i_{m(N-1)+k} = [(i_k + m(N/2-1)) \bmod N] + 1 \\ \{j_1, ...j_{N-1}\} = \{N, 2, N-1, 3, .., N/2+1\} \\ j_{m(N-1)+k} = [(j_k + m(N/2-1)] \bmod N] + 1 \end{cases}$$

Where $0 \leq m \leq N/2-1$ and $1 \leq k \leq N-1$. Each of the sequence $I_1 = \{i_1, ..., i_{N-1}\}$ and $J_1 = \{j_1, ..., j_{N-1}\}$ corresponds to a Hamiltonian path. Since K_{2n} is not Eulerian, a Hamiltonian path decomposition does not give a Eulerian cycle unless some edges are visited twice. In our setup, if an edge is supposed to be visited twice, then it must be subsequently traversed from both directions and the second walk should be treated as a flier. This algorithm generates m Hamiltonian paths for K_{2m}, and at the end of every Hamiltonian path, one should add a flier to reach the previous vertex to start the next Hamiltonian path. In an actual protocol, this is represented as node i transmits $\hat{h}^{(i)}_{i,k}$ and node j transmits $\hat{h}^{(j)}_{j,k}$ in two subsequent time slots. An example of a protocol for $N = 6$ designed using this algorithm is depicted below:

Lemma 5 (Upper bound of average staleness for K_{4k+2})**.** *The above protocol con-*

struction of $N = 4k + 2$ for all odd $k \geq 1$ results the average staleness less than or equal to $S^* + 1/2 - 1/N + 2/N^2$

The proof of which is shown in Appendix. However, applying this algorithm to design protocols for $N = 4k$ will not result in an L periodic protocol. So an algorithm based on Hamiltonian cycle decomposition is designed, and we present it as follows:

- step 1: Construct $N/2 - 1$ edge disjoint Hamiltonian cycles using the formula

$$\begin{cases} \{i_1, ..i_{N-1}\} = \{1, N-1, 2, N-2, 3, .., N/2\} \\ i_{mN+k} = [(i_k + m - 1) mod (N-1)] + 1 \\ i_{mN} = N \\ j_l = i_{[l-1 \bmod L]+1} \end{cases}$$

  Where $1 \leq m \leq N/2 - 2$, $1 \leq l \leq L$ and $1 \leq k \leq N - 1$.

- step 2: Circular shift the Hamiltonian cycles so that they all start from 1 (or any arbitrary number from $\{1, 2, .., N\}$). Append these Hamiltonian cycles to form a Eulerian cycle of some subgraph $G \in K_N$.

- step 3: Divide the Eulerian cycle of G into $N/2$ walks $W = \{w_1, ...w_{N/2}\}$ with length $N - 2$. Insert each $N/2$ dead end edges $\{e_{1,N-2}, e_{2,N-3}, ...e_{N-1,N}\}$ into one of the walks w_k.

Lemma 6 (Upper bound of average staleness for K_{2m})**.** *The above protocol construction of $N = 2m$ for all odd $m \geq 2$ results the average staleness asymptotically less than $S^* + 0.5$*

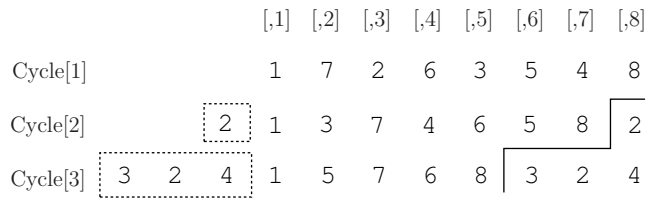


Figure 3.3: Step 1. and step 2. in constructing the protocol for K_8. Circular shifting the Hamiltonian cycles and align them results in a Eulerian tour of the induced sub-graph

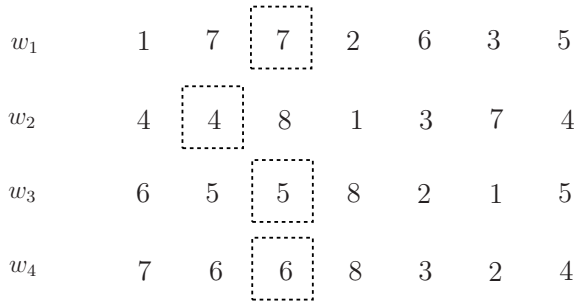


Figure 3.4: Step 3. in constructing the protocol for K_2m. First divide $m-1$ Hamiltonian cycles into m length $N-2$ walks w_i, then append m dead end edges $\{e_{1,N-2}, e_{2,N-3}, ..., e_{N-1,N}\}$ into these walks. It is easy to apprehend that dead end edges can be appended after $w_i[3]$ except w_2.

The proof of Lemma 3 is shown in Appendix. An example of the steps in constructing the protocol for K_8 is shown in Fig.3.3 and Fig.3.4.

Since K_{2m} can be decomposed into $m-1$ Hamiltonian cycles and a one-factor, connecting these cycles together will also form a cycle. The $N/2$ disjoint edges in the one-factor can be treated as discarding the in-degrees of length-2 cycles and replace them with fliers. Inserting them will not affect the cyclic nature of edge tour in our setup. From the resulting edge set, we can form a minimum worst case staleness protocol resulting the maximum average staleness less than $S^* + 0.5$. The proof of the upper bound of average staleness using these protocols is shown in appendix. Notice that this protocol also applies to the cases where $N = 4k + 2$.

3.3 Numerical results

The numerical results in simulating the global staleness for K_N, $0 \geq N \leq 200$ using the protocols indicates that the maximum staleness of these graphs achieves the theoretical lower bound, and the average staleness also stays within the theoretical bounds proposed. Fig. 3.5 indicates that the lower bounds of the worst case staleness are achieved through using the proposed protocols. Fig. 3.6 indicates that the achieved maximum average staleness using the proposed protocol also stay within the upper bound.

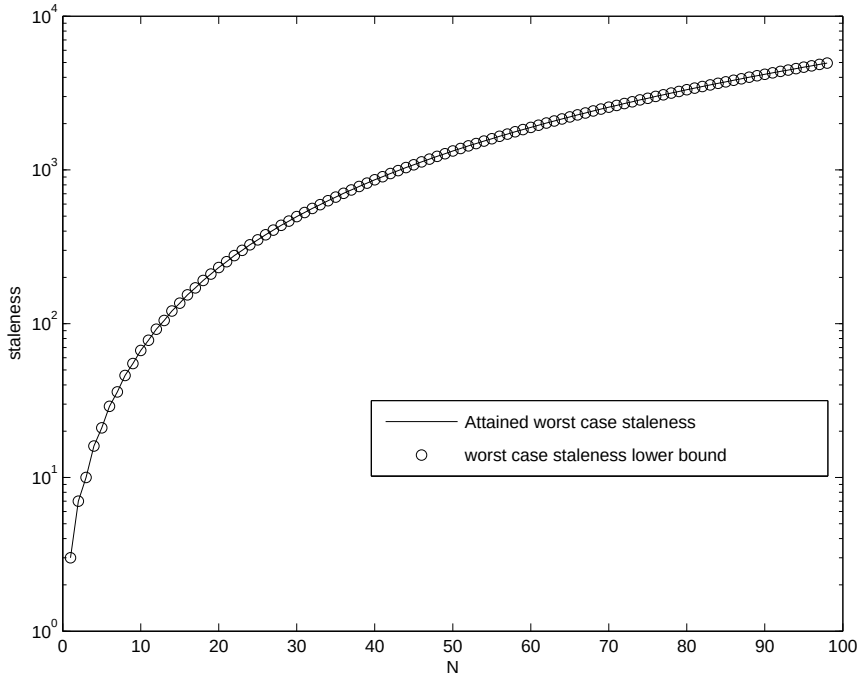


Figure 3.5: The attained worst case staleness using the proposed protocols and the worst case staleness lower bounds for K_N,$3 \leq N \leq 100$

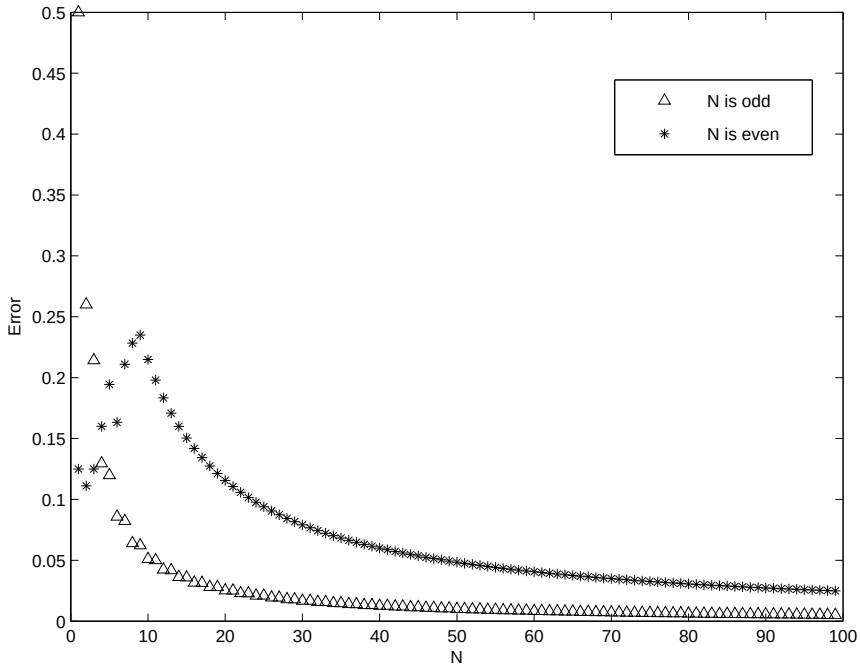


Figure 3.6: The difference between upper bounds of average staleness and the attained maximum average staleness using proposed protocols for K_N,$3 \leq N \leq 100$. The positive differences indicates that the maximum achieved average staleness stays within the upper bounds

Chapter 4

Minimum Staleness Protocol Design for Incomplete Graph

In this chapter, a more generalized model of the network having incomplete graph topology is concerned. The problem of minimum staleness CSI dissemination problem is transferred into finding minimum connected dominating set (CDS) in a graph. This chapter provides several approximate solutions for this problem and talks about realization issues. A randomized approach is also concerned when it is hard to find a near optimal solution.

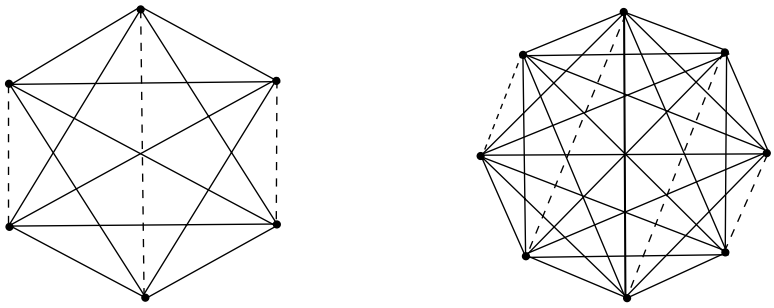
4.1 Problem Identification

Recall in the complete graph CSI dissemination problem, protocols designed from Eulerian tour are optimal in terms of worst case staleness based on the fact that any message only needs to be disseminated once to make sure that every other nodes in the network gets updated. In a incomplete graph, not dissemination can be done in just one time of transmission. Consider a source node u needs to disseminate

some estimate, the minimum time of transmission is yet bounded by the maximum distance $d(u, v)$, where $v \neq u$ and $v \in V(G)$. In these cases, the worst case staleness is no longer bounded by the number of links because some of the estimates may need to be transmitted multiple times to disseminated to every node of the network. A solution for a particular type of topology is shown in the following section.

4.1.1 Minimum Worst Case Staleness Protocols for $N-2$-Regular Graphs

A regular graph has the same degree on each vertex. A $N-2$-regular graph is a graph whose degree on every vertex is $N-2$, where N is the number of vertices. Notice that a $N-2$-regular graph cannot have an odd number of degree. An example of $N-2$-regular graphs is shown below in Fig. 4.1:



4-regular graph 6-regular graph

Figure 4.1: Examples of $N-2$-regular graphs

In $N-2$-regular graphs, the neighborhood of any vertex does not contain the vertex set $V(G)$, but the neighborhood of a pair of adjacent vertices does. So an estimate will not be disseminated to every node in just one time of transmission, but two time of transmission from a pair of neighboring vertices will. In finding a protocol to minimize worst case staleness, one would prefer vertices u and v to disseminate $\hat{h}_{u,v}$ respectively in a period since these vertices can possibly always

disseminate the instantaneous estimates if the protocol is properly designed.

Lemma 7 (Lower bound of worst case staleness in $N-2$-regular graphs)**.** *The minimum worst case staleness in $N-2$-regular graphs is greater or equal to $N(N-2)$*

Proof. There are $N(N-1)/2$ edges in a $N-2$-regular graph, each of the estimates needs to be transmitted at least twice to be disseminated to every vertex. The minimum period to disseminate all the estimates in terms of $N(N-2)/2$ edges is $N(N-2)$, so as the minimum worst case staleness □

Using the same format as the previous chapter: i_n, j_n stands for node i_n transmit estimate $\hat{h}_{i_n,j_n}$. Protocols achieve minimum worst case staleness in $N-2$-regular graphs is presented below:

- First $N(N-2)/2$ time slots

$$\begin{cases} \{i_1, ..i_{N-1}\} = \{1, N-1, 2, N-2, 3, .., N/2\} \\ i_{mN+k} = [(i_k + m - 1) mod (N-1)] + 1 \\ i_{mN} = N \\ j_l = i_{[l-1 \bmod L]+1} \end{cases}$$

  Where $1 \leq m \leq N/2-2$, $1 \leq l \leq N(N-2)/2$ and $1 \leq k \leq N-1$.

- Second $N(N-2)/2$ time slots

$$\begin{cases} i_n = i_{N(N-2)-n+1} \\ j_l = j_{N(N-2)-n+1} \end{cases}$$

  Where $N(N-2)/2+1 \leq n \leq N(N-2)$ and $N(N-2)/2+1 \leq l \leq N(N-2)$

The above formula generates protocols with period $N(N-2)$. During one period, node u and v transmit $\hat{h}_{u,v}$ once respectively so that every other node gets updated of $\hat{h}_{u,v}$. Since the nodes always transmit instantaneous estimates, the worst case staleness is the same as the period $N(N-2)$.

Notice that on a graph representation, the above formula generates a Hamiltonian decomposition of $N(N-2)$-regular graphs. The first half of the period is a Hamiltonian decomposition based Eulerian tour, and the second half of the period is doing the same Eulerian tour again in reversed direction.

4.1.2 Routing with CDS

The above example is a special case that the neighborhood of any pair of adjacent vertices spans the vertex set $V(G)$, so that designing a protocol which only transmit instantaneous estimates is possible. Consider the case when such assumption is not applicable, and one needs to disseminate, or broadcast some message from one node to the whole network with minimum number of transmission. A solution is building a maximum leaves spanning tree (MLST) whose root is the source node. Given the assumptions of the CSI problem settings, the number of transmission is equal to $N-L$, where L is the number of leaves in this spanning tree. Notice that finding a maximum number of leaf spanning tree is a duo of finding the minimum CDS, which is the set of non-leaf vertices of the spanning tree.

To obtain the spanning tree, or the CDS enabling the minimum time of transmission routing path, one can generate different MLST using different nodes as roots. However, despite the fact that minimum CDS in a graph is usually not unique, one can still just use a single fixed CDS to route. The difference is that if the source node is a leaf, the number of transmission will be increased by one compared to those who is not a leaf node.

4.2 Minimum CDS Algorithm and Protocol Design

In this section, a lower bound of the period of protocols is defined, and methods to approximate this lower bound is explored.

4.2.1 Lower Bound of Protocol Period

Consider a spanning tree is established as the spine of a Ad-hoc network. A dissemination of a message will be broadcast by every none-leaf node of the spanning tree to its neighbor to make sure that every node gets updated. Without loose of generality, it is assumed that once a round of dissemination completes (disseminating estimate of one link to every node), the next round of dissemination will start by some node broadcasting the estimate of an arbitrary link. In the best case, the number of transmission in one round of dissemination is the number of CDS, or the number of non-leaf nodes in the spanning tree, let it be $|C|$. Disseminating different types of estimates have various impact on the number of transmission:

- *Estimates of Links Between Dominating Nodes*

  The number of transmission is $|C|$. Since every node in the dominating set transmits once in one round of dissemination. Estimates of the links between them gets updated.

- *Estimates of Links Between a Leaf and a Dominating Node*

  The number of transmission is $|C|$. Since every node must do the transmission once in a period, including leaf nodes. So the estimate of the link between a leaf node and a dominating node can be formed and disseminated from the

dominating node. Dissemination of the estimate through CDS takes $|C|$ times of transmission.

- *Estimates of Links Between Leaf Nodes*

  Disseminating this type of estimates requires at least $|C| + 1$ times of transmission, since the source node of dissemination must be a leaf node.

Let M_{dd},M_{dl},M_{ll} be the number of links between dominating nodes, a dominating node and a leaf node, leaf nodes. The lower bound of the protocol period in an incomplete graph using fixed CDS routing is:

Lemma 8 (Trivial Lower bound)**.** *A trivial lower bound of protocol period using fixed CDS routingis* $(|C| + 1)(M_{ll}) + |C|(M_{dd} + M_{dl})$

Notice that $M_{ll} + M_{dl} + M_{dd} = L$, so the lower bound can be expressed as $L|C| + M_{ll}$.

However this lower bound is not always achievable. Consider disseminating the estimate of the link between two leaf nodes i, j and designate the source node disseminating the estimate to be j, then i must have transmitted at least once in previous time slots of the current period to make sure that this estimate was updated. Similar deductions can be made on a chain of nodes before i and their pairs. So the condition that M_{ll} links between leaf nodes will result in $(|C|+1)(M_{ll})$ times of transmission to disseminate these links is: all the links between leaf node will form a connected graph.

Further more, without loose of generality, the lower bound of protocol period using fixed CDS routing can be modified as:

Lemma 9. *The lower bound of protocol period using fixed CDS routing is* $(|C| + 1)(M_{ll}) + |C|(M_{dd} + M_{dl}) + C_l$, *or* $L|C| + M_{ll} + C_l$. *Where* C_l *is the number of connected graphs formed by the induced sub graph of the leaf nodes.*

Proof. Followed by Lemma 8. If the edges between leaf nodes forms more than one connected sub graph, then the dissemination process within one connected graph will not update the estimates of edges in other connected sub graphs. So at least one of the nodes in each connected sub graphs must initially transmit once (cannot be used as dissemination purpose) to update the estimates of the links within their sub graphs. This will give C_l times of additional transmissions than Lemma 8. □

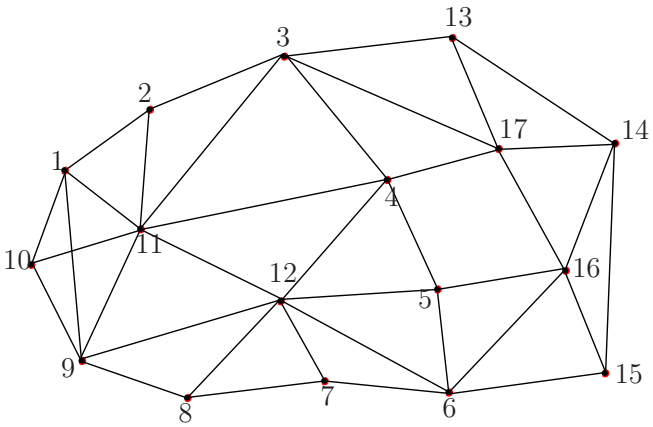
An example of the division of sub graphs formed by edges between leaf nodes is shown in Fig. 4.2

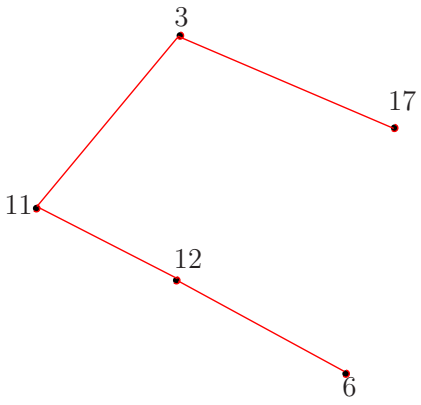
4.2.2 Scheduling in Dissemination of Estimates

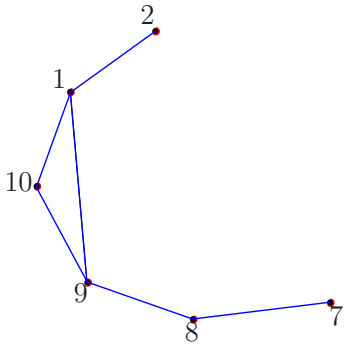
Even with the presence of a minimum CDS (or least CDS, with some luck), the protocol is yet to be completed. As described in previous sections, the links, or the edges can be categorized into three different types, denoted as e_{ll} (edges between leaf nodes), e_{ld} (edges between a leaf node and a dominating node) and e_{dd} (edges between dominating nodes). The scheme proposed below will address the dissemination of three types of estimates individually.

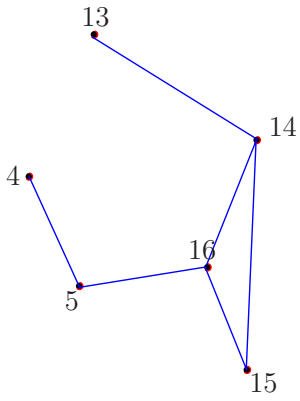
First let us consider the edges between dominating nodes. Not surprisingly, the algorithm constructing CDS will eventually form a maximum number leaves spanning tree (MLST), and the dominating set is also a subset of the spanning tree. Let T_d be the induced sub graph of the spanning tree T with leaves trimmed, and it is obvious that $V(T_d) = V(D)$, where D is the CDS of graph G. Edges of T_d is only a subset of e_{dd} however a definition of rank based on T_d is considered useful in scheduling the dissemination of e_{dd}.

According to Theorem 1.15 in [33], the center of a tree is either a single vertex or a pair of adjacent vertices, based on the fact that if one keeps removing the leaf









CDS induced sub graph

Leaf nodes induced sub graph

Figure 4.2: The CDS and connected sub graphs formed by edges between leaf nodes

nodes of a tree. The tree will end up having a single vertex or a pair of vertices. In Fig. 4.3 below, $rank(a) = 0$ and $rank(j) = 3$.

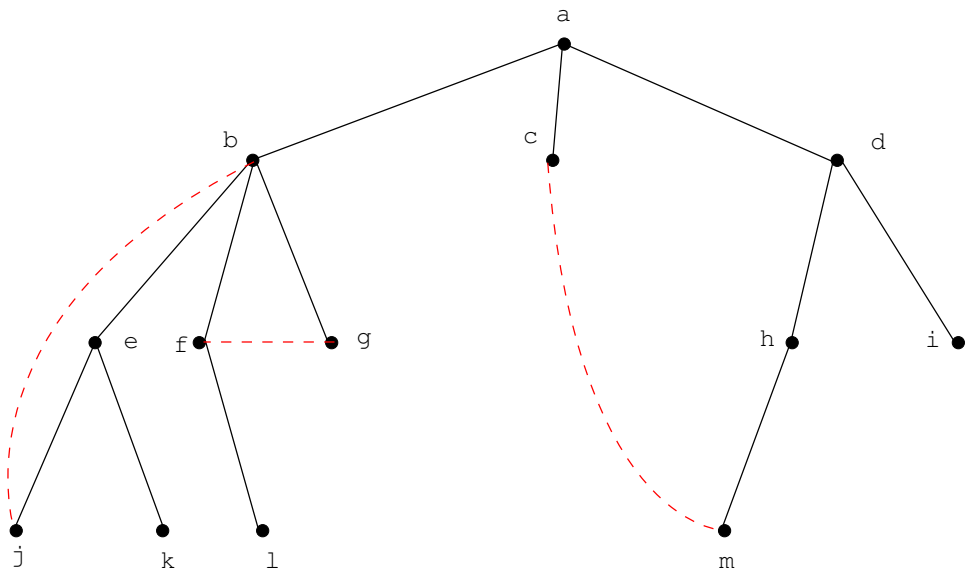


Figure 4.3: Spanning tree of CDS, red stroke lines are the edges between dominating nodes but not in the spanning tree

Before moving on to the scheduling of the dissemination of e_{dd}, there is yet another definition that needs to be attended.

Definition 3. *Let the center of T_d to be the root. The rank of a vertex u in T_d is defined as the number of offspring of u*

For example in Fig. 4.3, $rank(b) = 2$ and $rank(g) = 0$. The rank of a node u indicates the number of nodes whose dissemination is dependent on the transmission of u. Now consider the nodes having rank of 0. These nodes are the ones dominate the leaves, and in the process of disseminating an estimate, one of these nodes must be the last to transmit to make sure that all the leaf nodes get the disseminated estimate. Suppose in Fig. 4.3, $\hat{h}_{je}$ is selected to be disseminated by node e. To disseminate the estimate of e_{je}, every dominating node must transmit this estimate once and one of the rank 0 node $\{j, k, l, g, c, m, i\}$ must be the last to transmit, which indi-

cates that one of the estimates $\{\hat{h}_{jb}^{(b)}, \hat{h}_{je}^{(e)}, \hat{h}_{ek}^{(e)}, \hat{h}_{lf}^{(f)}, \hat{h}_{gf}^{(f)}, \hat{h}_{gb}^{(b)}, \hat{h}_{ca}^{(a)}, \hat{h}_{cm}^{(m)}, \hat{h}_{cm}^{(c)}, \hat{h}_{mh}^{(h)}, \hat{h}_{id}^{(d)}\}$ could be instantaneous estimate and thus preferable to be selected in the next round of dissemination.

Now define the vertex having the higher rank in an edge to be the **father** vertex, and the vertex with lower rank in the edge to be the **child** vertex. The rank of the child vertex of an edge determines the least staleness of the estimate represented by this edge could have when selected for dissemination (assuming the protocol designed does not increase the minimum possible period when disseminating E_{dd} type estimates). The rule of dissemination scheduling in order to minimize the staleness at the time some estimate is selected to be disseminated is: the **father** node of the previous disseminated edge cannot be the **child** node of the next disseminated edge. The reason for setting up this rule is, to minimize the staleness of the estimates the time they were to be selected to disseminate, the child node of the edge should be scheduled to be the last one to transmit during the previous round of dissemination, and the father node of the edge should be the initial node starting a new round of dissemination. Once the sequence of the edges need to be disseminated is determined, it is trivial to plan the routing for dissemination, as it is shown in the following example.

Example The routing of the transmission between disseminating e_{ej} and e_{ek} will be $\{e, j, b, f, i, g, a, c, d, b, m, i, k\}$ (they all transmit $\hat{h}_{ej}$). So at the end of $\hat{h}_{ej}$ dissemination, node k is the last one to transmit, making $\hat{h}_{ek}^{(e)}$ an instantaneous estimate.

Is it possible even through proper scheduling, it is eventually inevitable to select two edges of which the father node of the first edge will be the child node as the second edge? In fact with proper scheduling, the above situation will never happen. Firstly, if multiple edges share the same child node[1], one can schedule them to be

[1]It is not possible if the CDS induced sub graph is a tree. But if not, it is entirely possible that

subsequently disseminated because their father nodes must be different. For similar reasons, the multiple branches of a node will not cause a problem because these branches have the same father node thus there is no way that a father node of an edge is the child node of another, so these edges can be scheduled to be disseminated subsequently. The real concern is only the edges in the MLST and belong to the same main chain. Consider the worst case, that for every edge, there is another edge not eligible to be selected. [2] This problem is no more than seeking a Hamiltonian path on an directed graph which any vertex is connected to others in both direction except one of the directed edges is prohibited. To seek a Hamiltonian path, the least we can do is just use the reversed pairs of the unconnected directed paths as shown in Fig. 4.4

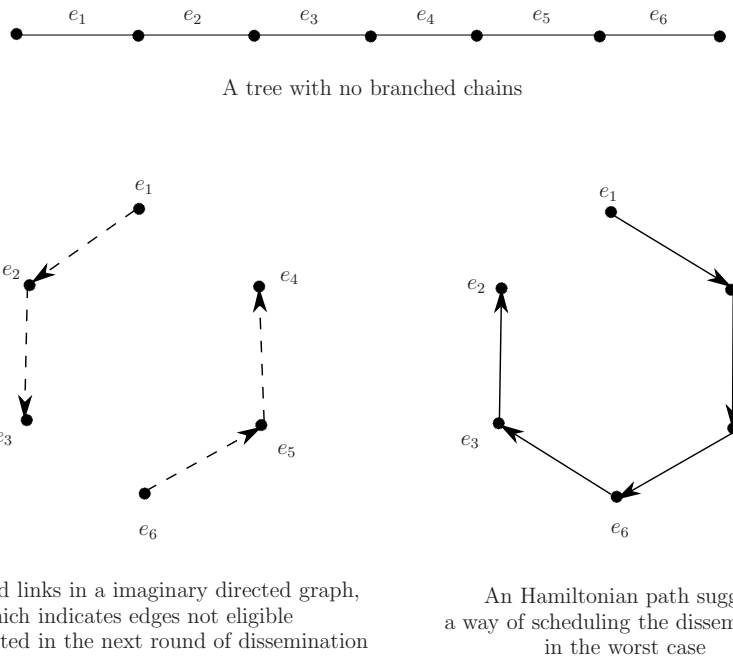


Figure 4.4: The topology for the worst case in scheduling the dissemination of edges on the spanning tree

the edges belong to the CDS induced sub graph but not to the MLST could share the same child node

[2]The topology for this worst case is a tree with only the main chain as shown in Fig. 4.4

Based on the above deductions, a traverse of edges from the root to the leaf is always permitted, which give rise to a stimuli of using depth first traverse or a width first traverse. Here a depth first traverse based scheduling algorithm is given. Suppose the CDS induced sub graph is D, while the CDS induced sub graph from MLST is T_d, we have:

ALGORITHM 1: SCHEDULING E_{dd} TYPE ESTIMATES

- **Step 1:** Start from the center of T_d. If the center is a pair of vertices, start from an arbitrary vertex of the two. Mark this vertex as u
- **Step 2:** Arbitrarily select an unused neighbor of u in T_d, mark it as v. Disseminate the edge whose child vertex is v, starting from u. If there are multiple edges in D whose child vertex is v, disseminate these edges subsequently while leaving the edge in T_d to be the last one to transmit in each round of dissemination. Mark v as used. Assign v as u.
- **Step 3:** Check for unused neighbors of u in T_d. If there are unused neighbors of u in T_d, go to Step 2, if not, go to Step 4.
- **Step 4:** Assign the parent node of u in T_d as u, go to Step 3. Terminate if there is no parent node to go.

Figure 4.5: Depth First Algorithm of Scheduling the Dissemination of E_{dd} Type Estimates

A similar definition of father/child nodes can also be applied in scheduling the dissemination of E_{ld} type estimates, except it simply defines the leaf node being the father node, and the dominating node being the child node. The rule to minimize the staleness becomes: the child nodes of the previously disseminated edge and the next disseminated edge cannot be the same. For those edges between a leaf node and a dominating node whose rank is nonzero, the minimum staleness when this

edge is selected to disseminate is the same as the rank of its child node. To properly schedule the order of dissemination so that this minimum staleness is attained, a depth first scheduling is presented below:

ALGORITHM 2: SCHEDULING E_{ld} TYPE ESTIMATES

- **Step 1:** Start with an arbitrary leaf node.
- **Step 2:** Examine the edges between the current leaf node and dominating nodes. Disseminate the the edge whose child node has the highest rank, then the edges whose child nodes is on the same main chain with lower ranks.
- **Step 3:** Go to the next main chain, start disseminating the edge with highest rank child node, then edges with lower rank child nodes. Repeat this step till every E_{ld} type edge linked to current leaf node is disseminated.
- **Step 4:** Go to the next leaf node and repeat step 2. Terminate if every leaf node is visited.

Figure 4.6: Algorithm of Scheduling the Dissemination of E_{ld} Type Estimates

By disseminating the edges whose child nodes on the same main chain in a rank descending order, the number of nodes transmitting after these child nodes in one round of dissemination will be the rank of the child nodes, so that the staleness when these edges are selected to disseminate will be minimized.

Now continue to the scheduling of E_{ll} type estimates. Since the leaf nodes do not transmit in the dissemination rounds except initializing E_{dl} or E_{ll} type estimates. Always using the instantaneous estimates in disseminating E_{ll} type is nearly impossible except allowing the leaf node transmit the training sequence in the start of each round of dissemination at the cost of one additional time step. Admitted, there are extreme cases when the number of E_{ll} type edges is much smaller than the

size of CDS which makes this approach reasonable, this thesis nevertheless present a scheme which does not require additional training steps. Consider a leaf node u transmit an estimate $\hat{h}_{uv}$ at time step n. Meanwhile a leaf node w, also a neighbor of u updates its estimate of $\hat{h}_{uw}^{(w)}$ at time step n. But w has to wait until the dissemination of $\hat{h}_{uv}$ completes to initiate the dissemination of $\hat{h}_{uw}$. After the dissemination of $\hat{h}_{uw}$, neighbors of w can start other rounds of dissemination. The scheduling of disseminating E_{ll} type estimates can be done by searching Eulerian trails (not cycles) in the induced sub graph by leaf nodes. If the sub graph does not contain a Eulerian tour (has more than 2 nodes with odd degree), the Eulerian tour can be done by modifying the graph as described in **section 3.1.2** to add "dead end paths" and fliers. If the induced sub graph contains more than one connected graphs, then disseminating E_{dl} type estimates can be inserted into the Eulerian tours of several connected graphs.

ALGORITHM 3: SCHEDULING E_{ll} TYPE ESTIMATES

- **Step 1:** Identifying the connected sub graphs in leaf nodes induced sub graph.
- **Step 2:** Start from a node having odd degree in an arbitrary connected sub graph. Disseminate edges in the order of a Eulerian trail. If the connected sub graph is not Eulerian, use the scheme in **section 3.1.2** to modify the graph.
- **Step 3:** After finishing Eulerian trail of current sub graph, insert a dominating-leaf edge dissemination whose leaf node is the one with odd degree in the next connected E_{ll} sub graph. Then start disseminating E_{ll} type estimates in the next connected sub graph by the order of Eulerian trail. Keep repeating **step 3** till all the E_{ll} type estimates are addressed.

Figure 4.7: Algorithm of Scheduling the Dissemination of E_{ld} Type Estimates

4.2.3 Finding Minimum CDS

Fining minimum CDS and finding MLST are equivalent. Although dominating sets can be found using boolean algebra, but the complexity is at least $O(2^v)$, where v is the number of vertices. Approximation algorithm of finding minimum CDS is well studied. J. Wu [30] and I. Stojmenovic [32] studied approximation algorithm of finding MCDS in wireless add-hoc networks, guaranteeing a $\frac{n}{2} - OPT$ approximation rate. Wan [37] proposed a distributed algorithm having $8 - OPT$ approximation rate. B. Das [31] proposed a routing scheme using MCDS found by algorithms adopted from S. Guha [38], which guarantees an approximation factor of $O(H(\delta)) - OPT$.

Apart from approximation algorithms, there are other algorithms trying to find MLST having leaves at least a fraction of the total number of nodes. Griggs J. [39] and Kleitman D J [40] proved that in a graph which has minimum degree of 3, fining a spanning tree with leaves at least $N/4 + 2$ is possible. Gao [41] proposed a 2-degree nodes reduction scheme so that the previous algorithms can be used on graphs containing 2-degree nodes. This these used the algorithm by Kleitman D J [40] to find the MLST of the graph in Fig. 4.2. [3].

4.3 Performance Analysis and Simulation Results

4.3.1 Worst Case Staleness Analysis

When disseminating an estimate through CDS, the estimate itself is rarely updated unless the dissemination goes through the same link as the estimate itself. Since the protocols are periodic, the maximum staleness of one particular estimate is the period plus the increase of staleness in the process of dissemination. In the previous

[3]This algorithm found the optimal solution of MLST in Fig. 4.2

introduced algorithm, the period of the protocol is:

$$L^* = (|C| + 1)(M_{ll} + M_{dl}) + |C|M_{dl} \tag{4.1}$$

Here is the maximum staleness in three types of estimates:

- E_{dd} type estimates: $L^* + |C| - 1$ since dissemination is initialized by instantaneous estimates, dissemination needs $|C|$ times of transmission, and the estimates update once in dissemination process.
- E_{dl} type estimates: $L^* + |C|$ since dissemination is initialized by transmitting instantaneous estimates. Dissemination needs $|C| + 1$ times of transmission. Estimates get updated once in dissemination process.
- E_{ll} type estimates: $L^* + |C| + 1 + (sgn(O/2 - 1))^+|C|$ where F is the maximum number of odd degree nodes in each connected sub graph belonging to the leaf nodes induced sub graph.

So based on the deductions above, the upper bound guaranteed by the algorithm is:

$$S_{upper} = (|C| + 1)(M_{ll} + M_{dl}) + |C|M_{dd} + |C| + 1 + 2(sgn(O/2 - 1))^+|C| \tag{4.2}$$

Recall the trivial lower bound of worst case staleness in Lemma 8:

$$S_{LB} = (|C| + 1)M_{ll} + |C|(M_{dl} + M_{dd}) = |C|L + M_{ll}(|C| - 1)^+ + |C| - 1 \tag{4.3}$$

or

$$S_{LB} = (|C| + 1)M_{dl} + |C|M_{dd} + |C| - 1 = |C|L + M_{dl} + |C| - 1 \tag{4.4}$$

In cases when the original graph has no loop.

For the algorithm guaranteed upper bound we have:

$$\begin{aligned} S_{upper} &= (|C|+1)(M_{ll}+M_{dl}) + |C|M_{dd} + |C| + 1 + 2(sgn(O/2-1))^{+}|C| \\ &\leq (|C|+1)(M_{ll}+M_{dl}) + |C|M_{dd} + M_{dd} - 1 + 1 + 2(sgn(O/2-1))^{+}|C| \\ &= (|C|+1)L + 2(sgn(O/2-1))^{+}|C| \\ &\leq (|C|+1)L + 2|C| \\ &\leq S_{LB} + L - 2|C| \end{aligned} \tag{4.5}$$

Consider there are loops in the original graph and the leaves induced sub graph has no Eulerian trail. In this case, $N \geq 5$. Or:

$$S_{super} \leq S_{LB} + L \tag{4.6}$$

When leaves induced sub graphs are Eulerian, and in this case $N \geq 4$ or:

$$S_{upper} = (|C|+1)M_{dl} + |C|M_{dd} + |C| - 1 = S_{LB} \tag{4.7}$$

When there are no loops in original graph.

4.3.2 Numerical Results

The following numerical result is done by applying the proposed CSI dissemination algorithm to the network having topology as Fig. 4.2. This graph has 37 edges, 5 out of which are E_{dd} type, 22 of which are E_{dl} type and 10 of which are E_{ll} type. The CDS is found by using the algorithm proposed in [40]. The leaf induced sub graph has two connected sub graphs and none of them are Eulerian. The simulation

assume perfect estimate at initial state.

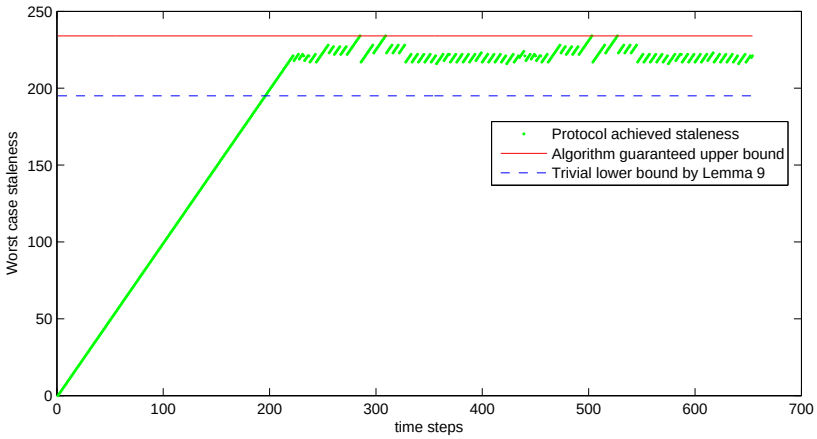


Figure 4.8: Simulation result of CSI dissemination problem in networks with topology as Fig. 4.2

Chapter 5

Conclusion

The CSI dissemination is proposed with idealized constraints and assumptions. Though an open problem as it is, the CSI dissemination problem for networks with complete graph topology with current constraint is solved by the method proposed in the thesis, and the solution is optimal in terms of minimizing worst case staleness and is near optimal in terms of minimizing average staleness. The thesis also proposed a near optimal algorithm solving CSI dissemination problem in networks with incomplete graph topology in terms of minimizing worst case staleness, and the approximation factor of such algorithm is given.

While a solid formulated approach generating protocols for complete graph is given in Chapter 3, the algorithms in Chapter 4 somehow rely on approximation or heuristic algorithms finding minimum CDS. Admitted, there is no precise algorithm finding minimum CDS by far thus it is a NP-hard problem, but once an minimum CDS, or the least CDS is found, this algorithm is an efficient way to exploit CDS to obtain a near optimal solution. Numerical results in Chapter 3 and Chapter 4 verified the proposed schemes of building protocols.

Since the CSI dissemination problem can be a new field of study, future works

can be done in various assumptions and settings: consensus estimation compatible protocols allow the interchange of the estimates made by two nodes estimating the link between them, while maintaining low staleness at the same time; self organized scheme to allow protocol combating time-variant topology; randomized schemes not requiring prior knowledge of the topology. Non-ideal estimations can also be considered to evaluate the estimation performance under different protocols.

Appendix A

Appendix: Proofs

A.1 Proof of Lemma 3

Proof. The protocol constructed using the above method consists of m length $N-1$ sequence of the nodes for transmission $J_1 = \{j_1, ...j_{N-1}\} = \{N, 2, N-1, 3, .., N/2+1\}$. Each length $N-1$ transmission sequences consists of $N-1$ distinct nodes. Notice the fact that $J_{k+1} = [J_1 + k(N/2-1)] \bmod N$, therefore, the $(N-2)th$ element j_{kN-2} in sequence J_k must not be in sequence J_{k+1}. Consider the distance between current time slot and the head of the sequence S_α is x. Since the last x nodes used for transmission are distinct, resulting the staleness of x groups of instantaneous estimates to be $\{0, 1, ..., x-1\}$. Let the last nodes used for transmission in $S_{\alpha-1}$ be identical to the first x nodes in S_α so that the remaining $N-x$ nodes will have the maximum sum of staleness. So the the sum of staleness of the instantaneous estimates is $x(x-1)/2 + [(2x+1) + (N+x-2)](N-x-2)/2 + (x+1) + (N-x)$, weighed by $N-1$. Notice the last two addend are results from the $(N-2)th$ element in $S_{\alpha-1}$ which is distinct from the set of the first x elements in S_α, and the $(N-2)th$ element in $S_{\alpha-2}$ which is distinct from the $N-1$ elements in $S_{\alpha-1}$.

Taking the derivative and let it be zero yields $x = N/2 - 1$. Use the integer solution of $x = N/2$ yields the maximum average staleness $S^* + 1/2 - 1/N + 2/N^2$ □

A.2 Proof of Lemma 4

Proof. Consider the node transmission sequence defined by step 1 and step 2: Apart from the second sequence, they are generated by taking modulo and left circular shifting the previous sequence by 2. So the general expression can be written as $i_{mN+k} = [(i_{[(k+2m-2) \bmod (N)]+1} + m - 1) \bmod (N-1)] + 1$, with $\{i_1, ..i_{N-1}\} = \{1, N-1, 2, N-2, 3, .., N/2\}$ and $i_N = N$, $i_{(m+1)N-2m+1} = N$ where $1 \leq m \leq N/2 - 2$, $1 \leq l \leq L$ and $1 \leq k \leq N - 1$. Step 3 requires inserting $N/2$ disjoint edges $\{(i_1, i_{N-2}), (i_2, i_{N-3}), ...(i_{N/2-1}, i_{N/2}), (i_{N-1}, i_N)\}$ into $N/2$ evenly divided walks of length $m - 2$. Notice that $\{i_2, i_{N-1}, i_{2N-2}, i_{3N-4}, ...i_{(N/2-1)*(N-2)+2)}\}$ is the following sequence $\{N-1, N/2, N/2+1, ...N-2\}$, which forms a "base" to conjunct the $N/2$ disjoint edges. A deterministic realization is inserting the $N-2$ disjoint edges at the head of these vertices and add flyers as previously mentioned. The complete protocol is a L period transmission sequence consisting of one length $N + 2$ sequence and $N/2 - 2$ length $N + 1$ sequences, each containing a complete set $[N]$. This division is a result from the original $N/2 - 1$ Hamiltonian cycles. Considering different correlations between these sequences, the search for the upper bound of the average staleness given by the above protocols can be done in the following three regions:

- *odd-to-even* Current time slot resides in a length $n + 2$ sequence containing a Hamiltonian cycle, previous sequence has length $n + 1$.

- *even-to-odd* Current time slot resides in a length $n + 1$ sequence containing a Hamiltonian cycle, previous sequence has length $n + 2$.

- *odd-to-odd* Current time slot resides in a length $n+1$ sequence containing a Hamiltonian cycle, previous sequence has length $n+1$.

Since given our setups, the length $n+2$ sequence and the previous length $n+1$ sequence has the least correlation while the length $n+1$ sequence and the previous length $n+2$ sequence has the most correlation. We consider the odd-to-even case, where the two sequences has the least correlation. As depicted in Fig. A.1, when $3 \leq x \leq N-1$, the length x sequence contains at least 1, $N-1$ and $N-2$, which correspond to the $N+1$st, $N-1$st and $N-2$nd element in sequence N_2. Notice that the $x-2$nd element in sequence N_1 is from the one factor. Without loose of generality, we assume that the $x-2$nd element is a duplicate of some previous element in sequence x. Therefore, the sum of staleness in sequence x is $x-1+(0+x-3)(x-2)/2$. Since there are only $x-1$ distinct element in sequence x, and three of which appears at the end of sequence N_2, so there are at most $x-4$ distinct element in the first $x-4$ element of N_2 which are also in sequence x. Consider there is one element in N_2 generated by inserting one-factor, we assume this element is a duplicate of some element in sequence x since we are trying to maximize the sum of staleness. In that case, we modify the argument that there are $x-4$ distinct elements in the first $x-3$ elements in sequence N_2, and they are all duplicates of elements in x. Hence the sum of staleness in sequence N_2 is $[(2x-3)+(N+x-4)](N-x)/2+N+x-1$. Adding the two sums of staleness together and take the derivative of x yields $-2x+3+N$. Let the derivative be zero gives $x = N/2+3/2$. Notice that x should be an integer, thus using $x = N/2+1$ or $x = N/2+2$ will give us the upper bound of the sum of the average staleness in a given time, which is $S^*1/2 - 1/N + 6/N^2$. We can see that this bound is asymptotically smaller than $S^* + 0.5$. □

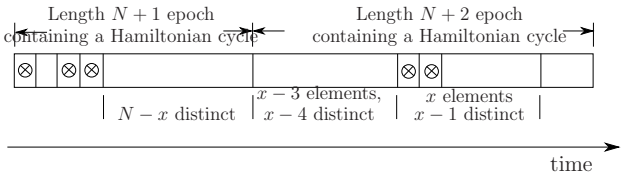


Figure A.1: Odd-to-even case using the protocol construction menthioned earlier. The circular crossing indicates that this node number is identical to some nodes that latter did the transmission, thus these nodes have no contribution in calculating the global staleness

Appendix B

Code Listings

B.1 Code Listings of complete graph CSI dissemination

B.1.1 Function Calculating Staleness

```
1  function [l_bound, u_bound, min_staleness, max_staleness]=
       stalenessCalcOddEven(N)
2  % Computes and prints staleness results to screen.
3  % Number of nodes N must be odd.
4
5  % Version history:
6  %   AGK - v1.0 30-sep-2012 - initial version
7  %   AGK - v1.1 01-oct-2012 - fixed bug in upper bound
8  %    Wenmin - OddEven
9  %    Wenmin - Random
10
11 % determine optimal protocol
12 protocol = optimalProtocolOddEven_v2(N); % 2-column format
```

```
13 protocol = [protocol(:,1) min(protocol,[],2) max(protocol,[],2)]; % 3-
       column format
14 protocol = [protocol;protocol]; % do it twice
15
16 % initialize timestamps... only use upper triangular portion of matrix,
       -Inf means "no information", +Inf is unused part of matrix
17 L=(N^2-N)/2;
18 timestamps=zeros(N,N,N);
19 junk=toeplitz(Inf*ones(N,1),[Inf -Inf*ones(1,N-1)]);
20 for i=1:N
21     timestamps(:,:,i)=junk;
22 end
23
24 % loop through protocol, and compute average and max staleness
25 maxstaleness=zeros(size(protocol,1),1);
26 avgstaleness=zeros(size(protocol,1),1);
27 for i=1:size(protocol,1)
28     % do dissemination first
29     disseminatedtime=timestamps(protocol(i,2),protocol(i,3),protocol(i
           ,1));          % timestamp of value that's being disseminated
30     idx=find(disseminatedtime > timestamps(protocol(i,2),protocol(i,3)
           ,:));          % list of nodes who will update to this new time
31     timestamps(protocol(i,2),protocol(i,3),idx)=disseminatedtime*ones(
           size(idx));  % do the update
32
33     % then do observation/direct estimation (some disseminated values
34     %    may get overwritten, but direct observations are always
           freshest)
35     disseminatedtime=i;
36     for j=1:protocol(i,1)-1
```

```
37            timestamps(j,protocol(i,1),j)=disseminatedtime;  % do the
                  update
38        end
39        for j=protocol(i,1)+1:N
40            timestamps(protocol(i,1),j,j)=disseminatedtime;  % do the
                  update
41        end
42
43        % calculate max/avg staleness over all N nodes over all (N^2-N)/2
              links
44        maxstaleness(i)=max(max(max(i-timestamps)));
45        avgstaleness(i)=sum(sum(triu(sum(i-timestamps,3),1)))/L/N;
46
47    end
48
49    % calculate average staleness bounds
50    l_bound=(N^3-3*N^2+8*N-8)/(4*N);
51    %penalty=(1/2*N^2-N-3/2)/N^2;
52    %penalty = (N^2 - 1)/2/N^2;
53    if mod(N,2)
54        u_bound=l_bound+1/2;
55    elseif mod(N,4)
56        l_bound=(N-2)/N/(N-1)+l_bound;
57        u_bound=l_bound+1/2;
58    else
59        l_bound=(N-2)/N/(N-1)+l_bound;
60        u_bound=l_bound+1/2;
61    end
62    max_staleness = max(avgstaleness(end-L+1:end));
63    min_staleness = min(avgstaleness(end-L+1:end));
```

```
64 %compute staleness achieved, and lower/upper bounds, and display
       results
65 disp(' ')
66 disp([' ---------- Results for N = ' num2str(N) '
       ----------'])
67 disp(['          Minimum worst-case staleness achieved: ' num2str(min(
       maxstaleness(end-L+1:end)),'%.4f')])
68 disp(['          Maximum worst-case staleness achieved: ' num2str(max(
       maxstaleness(end-L+1:end)),'%.4f')])
69 disp(' ')
70 disp(['Average staleness achieved, averaged over time: ' num2str(mean(
       avgstaleness(end-L+1:end)),'%.4f')])
71 disp(['             Minimum average staleness achieved: ' num2str(min(
       avgstaleness(end-L+1:end)),'%.4f')])
72 disp(['             Maximum average staleness achieved: ' num2str(
       max_staleness,'%.4f')])
73 disp(['Theoretical bound: ' num2str(l_bound,'%.4f') ' <= avg staleness
       <= ' num2str(u_bound,'%.4f')])
74 disp(' ')
75 % savefile = 'avg_staleness';
76 %save(savefile,'l_bound','max_staleness');
```

B.1.2 Function Generating Optimal Protocols

```
1 function strategy=optimalProtocolOddEven_v2(N)
2 switch mod(N,2)
3     case 1
4         n=floor(N/2);
5         L=(N^2-N)/2;
6         strategy=zeros((N^2-N)/2,2);
7         %hamiltonian cycle decomposition of N odd
```

```
8           base=[1:n;2*n:-1:n+1];
9           base=base(:)';
10          for i=1:n
11              strategy((1:N)+N*(i-1),2)=[mod(base+i-2,2*n)+1 N]';
12          end
13          strategy(1:L-1,1)=strategy(2:L,2);
14          strategy(L,1)=1;
15      otherwise
16
17              %Kn can be decomposite into n/2-1 hamiltonian cycles
18              %and n/2 disjoint pairs
19              n = N/2 - 1;
20              strategy = zeros(n*N,2);
21              base = [2*n:-1:n+1;1:n];
22              base = base(:)';
23              base = base + 1;
24              base = [1 base];
25              for i = 1:n
26                  temp = zeros(1,N);
27                  temp(1:N-1) = mod(base+i-2,N-1)+1;
28                  temp(N) = N;
29                  if find(temp == 1) ~= 1
30                      strategy((1:N) + N*(i-1),1) = [temp(find(temp == 1)
                            :end)';temp(1:find(temp == 1)-1)'];
31                      %strategy((1:N) + N*(i-1),1)'
32                  else
33                      strategy((1:N) + N*(i-1),1) = temp';
34                      %strategy((1:N) + N*(i-1),1)'
35                  end
36              end
37              strategy(:,2) = [strategy(2:end,1);strategy(1,1)];
```

```
38
39              % revert the strategy, due to some historical reasons
40              strategy(:,[1 2]) = strategy(:,[2 1]);
41
42              %auxilary stores the n/2 disjoint pairs
43 %              auxilary = [1:N/2-1;N-2:-1:N/2];
44 %              auxilary = auxilary';
45 %              auxilary = [auxilary;[N-1 N]];
46              shift = 2*ones(N/2,1);
47              shift(2) = 1;
48              for j = 1:1:N/2
49 %                  w = strategy(j*(N-1),2);
50                    i = shift(j);
51                    %add flyers to the cycle
52 %                    if any(any(ismember(auxilary,strategy((j-1)*(N-1)
       +i,2))))
53 %                        auxilary(mod(find(auxilary == strategy((j-1)
       *(N-1)+i,2)) - 1,size(auxilary,1)) + 1,:) = [];
54                        w = strategy((j-1)*(N-1)+i,2);
55                        if w == N
56                            v = N-1;
57                        elseif w == N-1
58                            v = N;
59                        else
60                            v = N - w -1;
61                        end
62                        % the eulerian cycle is divided into n/2 parts,
63                        % plug in one pair whenever possible in every
                            part
64                        if ((j-1)*(N-1) + i) == 1
65                            strategy = [[v w];strategy];
```

```
66                          else
67                              strategy = [strategy(1:(j-1)*(N-1)+i-1,:);[
                                    v w];strategy((j-1)*(N-1)+i:end,:)];
68                          end
69
70 %                      if w ~= N/2
71 %                          v = N - w;
72 %                          strategy = [strategy(1:j*(N-1)-1,:); [v w];
       strategy(j*(N-1):end,:)];
73 %                      else
74 %                          v = N;
75 %                          strategy = [strategy(1:j*(N-1)-1,:); [v w];
       strategy(j*(N-1):end,:)];
76 %                      end
77              end
78              %strategy = [strategy(2:end,:);strategy(1,:)];
79 end
80 end
```

B.2 Code Listings of minimum CDS CSI Dissemination

B.2.1 Function Calculating Staleness

```
1 function stalenessCalcDecompose(edge)
2 % Computes and prints staleness results to screen.
3 % Number of nodes N must be odd.
4 % G is the adjacent matrix
5
6 % Version history:
```

```
7  %    AGK - v1.0 30-sep-2012 - initial version
8  %    AGK - v1.1 01-oct-2012 - fixed bug in upper bound
9  %    Wenmin - General Complete graph
10 %    Wenmin - Random Protocol
11 %    Wenmin - General Incomplete Graph
12
13 % determine optimal protocol
14 G = zeros(max(max(edge)),max(max(edge)));
15 for i = 1:size(edge,1);
16     G(edge(i,1),edge(i,2)) = 1;
17     G(edge(i,2),edge(i,1)) = 1;
18 end
19 N = size(G,1);
20 %generate Maximum number leaf spanning tree
21 tree = MLST(edge);
22 [edge_list, Td_edge, none_leaf] = graph_decompose(edge,tree);
23 % randomly pick an edge
24
25 protocol = ProtocolDecompose(edge_list,Td_edge,none_leaf,tree);
26 protocol = [protocol; protocol; protocol];
27 %protocol = [2 1 2;1 1 2;5 1 5;1 1 5;4 1 4;1 1 4;3 1 3;1 1 3;2 1 2;3 2
       3;1 2 3;4 1 4;3 3 4;1 3 4;4 1 4;5 4 5;1 4 5];
28
29 % initialize timestamps... only use upper triangular portion of matrix,
       -Inf means "no information", +Inf is unused part of matrix
30
31 L=size(edge,1);
32 timestamps=zeros(N,N,N);
33 junk=toeplitz(0*ones(N,1),[0 0*ones(1,N-1)]);
34 for i=1:N
35     timestamps(:,:,i)=junk;
```

```
36 end
37
38 % loop through protocol, and compute average and max staleness
39 maxstaleness=zeros(size(protocol,1),1);
40 avgstaleness=zeros(size(protocol,1),1);
41 for i=1:size(protocol,1)
42     % do dissemination first
43     disseminatedtime=timestamps(protocol(i,2),protocol(i,3),protocol(i
           ,1)); % timestamp of value that's being disseminated
44     idx=find(disseminatedtime > timestamps(protocol(i,2),protocol(i,3)
           ,:)); % list of nodes who will update to this new time
45     inx = find(G(protocol(i,1),:)>0);
46     indx = intersect(idx,inx);
47     timestamps(protocol(i,2),protocol(i,3),indx)=disseminatedtime*ones(
           size(indx)); % do the update
48
49     % then do observation/direct estimation (some disseminated values
50     %   may get overwritten, but direct observations are always
           freshest)
51     disseminatedtime=i;
52     for j=1:protocol(i,1)-1
53         if G(protocol(i,1),j)
54             timestamps(j,protocol(i,1),j)=disseminatedtime; % do the
                   update
55         end
56     end
57     for j=protocol(i,1)+1:N
58         if G(protocol(i,1),j)
59             timestamps(protocol(i,1),j,j)=disseminatedtime; % do the
                   update
60         end
```

```
61      end
62
63      % calculate max/avg staleness over all N nodes over all (N^2-N)/2
            links
64      maxstaleness(i)=max(max(max(i-timestamps(timestamps ~= 0))));
65      avgstaleness(i)=sum(sum(sum(i-timestamps(timestamps ~= 0),3),1))/L/
            N;
66
67  end
68
69  % calculate average staleness bounds
70  l_bound=(N^3-3*N^2+8*N-8)/(4*N);
71  penalty=(1/2*N^2-N-3/2)/N^2;
72  u_bound=l_bound+penalty;
73  max_staleness = max(avgstaleness(end-L+1:end));
74
75  % compute staleness achieved, and lower/upper bounds, and display
        results
76
77  disp(' ')
78  disp(['---------- Results for N = ' num2str(N) '
        ----------------------------------------'])
79  disp(['          Minimum worst-case staleness achieved: ' num2str(min(
        maxstaleness(end-L+1:end)),'%.4f')])
80  disp(['          Maximum worst-case staleness achieved: ' num2str(max(
        maxstaleness(end-L+1:end)),'%.4f')])
81  disp(' ')
82  disp(['Average staleness achieved, averaged over time: ' num2str(mean(
        avgstaleness(end-L+1:end)),'%.4f')])
83  disp(['             Minimum average staleness achieved: ' num2str(min(
        avgstaleness(end-L+1:end)),'%.4f')])
```

```
84 disp(['                Maximum average staleness achieved: ' num2str(
       max_staleness,'%.4f')])
85 %disp(['Theoretical bound: ' num2str(l_bound,'%.4f') ' <= avg staleness
       <= ' num2str(u_bound,'%.4f')])
86 disp(' ')
```

B.2.2 Function Scheduling Dissemination Edge Sequences

```
 1 function [edge_list, Td_edge, none_leaf] = graph_decompose(
       original_graph,tree)
 2 %decompose the graph into: dominating set induced subgraph, MLST, leaf
 3 %induced subgraph, leaf-dominating edges
 4
 5 %edge list is the output of the scheduling of the edges
 6 edge_list = [];
 7
 8 %none-leaf vertex set
 9 edge = reshape(tree,2*size(tree,1),1);
10 [n,bin] = histc(edge,unique(edge));
11 multiple = find(n>1);
12 index = ismember(bin,multiple);
13 none_leaf = unique(edge(index));
14 leaves = setxor(unique(original_graph),none_leaf);
15
16 %construct original graph
17 g = graph(original_graph);
18
19 %construct MLST graph
20 T = graph(tree);
21 Td = graph;
22 induce(Td,T,none_leaf);
```

```
23 Td_edge = edges(Td);
24 Td_edge = reshape(Td_edge,size(Td_edge,1)*2,1);
25 Td_edge = none_leaf(Td_edge);
26 Td_edge = reshape(Td_edge,size(Td_edge,1)/2,2);
27
28 %cds induced graph
29 cds = graph;
30 induce(cds,g,none_leaf);
31 cds_edge = edges(cds);
32 cds_edge = reshape(cds_edge,size(cds_edge,1)*2,1);
33 cds_edge = none_leaf(cds_edge);
34 cds_edge = reshape(cds_edge,size(cds_edge,1)/2,2);
35
36 %leaf induced graph
37 ell = graph;
38 induce(ell,g,leaves);
39 leaf_edge = edges(ell);
40 leaf_edge = reshape(leaf_edge,size(leaf_edge,1)*2,1);
41 leaf_edge = leaves(leaf_edge);
42 leaf_edge = reshape(leaf_edge,size(leaf_edge,1)/2,2);
43 leaf_vertex = unique(leaf_edge);
44
45 %edge list of edl
46 edl_edge = setdiff(original_graph,cds_edge,'rows');
47 edl_edge = setdiff(edl_edge,fliplr(cds_edge),'rows');
48 edl_edge = setdiff(edl_edge,leaf_edge,'rows');
49 edl_edge = setdiff(edl_edge,fliplr(leaf_edge),'rows');
50
51 %neighboring matrix of Td
52 %search for the center of Td
53 G = zeros(max(max(original_graph)),max(max(original_graph)));
```

```
54 for i = 1:size(tree,1);
55     G(tree(i,1),tree(i,2)) = 1;
56     G(tree(i,2),tree(i,1)) = 1;
57 end
58
59 count = sum(G);
60 while length(find(count > 1)) > 1
61     index = count == 1;
62     G(index,:) = 0;
63     G(:,index) = 0;
64     count = sum(G);
65 end
66
67 %center of Td
68 center = find(count > 1);
69
70 %start the scheduling of E_dd type estimates start with the center, go
71 %depth first on Td. If an vertex is a child of multiple E_dd, start
       with
72 %those are not father-offspring pattern, put the father-offspring edge
       to
73 %be the last one to disseminate
74
75 %neighboring matrix of original
76 G_origin = zeros(max(max(original_graph)),max(max(original_graph)));
77 for i = 1:size(original_graph,1);
78     G_origin(original_graph(i,1),original_graph(i,2)) = 1;
79     G_origin(original_graph(i,2),original_graph(i,1)) = 1;
80 end
81
82 %neighboring matrix of Td
```

```
G = zeros(max(max(original_graph)),max(max(original_graph)));
for i = 1:size(Td_edge,1);
    G(Td_edge(i,1),Td_edge(i,2)) = 1;
    G(Td_edge(i,2),Td_edge(i,1)) = 1;
end

%neighboring matrix of cds
G_cds = zeros(max(max(original_graph)),max(max(original_graph)));
for i = 1:size(cds_edge,1);
    G_cds(cds_edge(i,1),cds_edge(i,2)) = 1;
    G_cds(cds_edge(i,2),cds_edge(i,1)) = 1;
end

%neighboring matrix of leaf induced sub graph
G_leaf = zeros(max(max(original_graph)),max(max(original_graph)));
for i = 1:size(leaf_edge,1);
    G_leaf(leaf_edge(i,1),leaf_edge(i,2)) = 1;
    G_leaf(leaf_edge(i,2),leaf_edge(i,1)) = 1;
end

% %neighboring matrix of Edl
G_edl = G_origin - G_cds - G_leaf;

%tags for all the vertices
used = zeros(size(unique(original_graph),1),1);
father = zeros(size(unique(original_graph),1),1);
u = center;
while size(edge_list,1) < size(cds_edge,1)
    %neighbors of u
    neigh = find(G(u,:)>0);
    except = find(used > 0);
```

```
114     %do not use used node, do not use the father node
115     neigh = setdiff(neigh,except);
116     neigh = setdiff(neigh,father(u));
117     if ~isempty(neigh)
118         v = neigh(1);
119         used(v) = 1;
120         father(v) = u;
121         subgraph = G - G_cds;
122         neigh_v = find(subgraph(v,:)>0);
123         if isempty(neigh_v)
124             %if neighbor of child node empty, just disseminate current
                    edge
125             edge_list = [edge_list; [u v]];
126         else
127             %if neighbor of child node not empty, disseminate those
                    edges not
128             %in Td and then current edge
129             for i = 1:length(neigh_v)
130                 if ~used(neigh_v(i))
131                     edge_list = [edge_list;[neigh_v(i) v]];
132                 end
133             end
134             edge_list = [edge_list; [u v]];
135         end
136         %after dissemination, assign child as the father and keep on
137         u = v;
138     else
139         %if there is no neighbor, trace back
140         u = father(u);
141     end
142 end
```

```
143
144 neigh_of_edd_tail = find(G_origin(edge_list(end),:)>0);
145 start_of_ell = intersect(neigh_of_edd_tail,unique(leaf_edge));
146
147 if ~isempty(start_of_ell)
148     u = start_of_ell(1);
149     v = edge_list(end);
150     edge_list = [edge_list; [u v]];
151     edl_edge = setdiff(edl_edge,[u v],'rows');
152     edl_edge = setdiff(edl_edge,[v u],'rows');
153     G_edl(u,v) = 0;
154     G_edl(v,u) = 0;
155 end
156
157 %define the number of connected subgraph induced by leaf vertices
158 candidate = leaf_vertex;
159 subgraph = zeros(max(max(original_graph)),max(max(original_graph)));
160 k = 0;
161 while ~isempty(candidate)
162     k = k + 1;
163    Connected_sub = CommIdentify(G_leaf,u);
164    subgraph(k,1:length(Connected_sub)) = Connected_sub;
165    candidate = setdiff(candidate,Connected_sub);
166    if ~isempty(candidate)
167        u = candidate(1);
168    else
169        break;
170    end
171 end
172
173 %vertex set of the sub graphs
```

```
174 subgraph = subgraph(sum(subgraph,2)>0,:);
175 G_sub = zeros(max(max(original_graph)),max(max(original_graph)));
176 %decompose leaf induced sub graph into several connected subgraph, and
        find
177 %their eulerian trails respectively
178 for i = 1:size(subgraph,1)
179     G_sub(subgraph(i,subgraph(i,:) > 0),:) = G_leaf(subgraph(i,subgraph
            (i,:) > 0),:);
180     G_sub(:,subgraph(i,subgraph(i,:) > 0)) = G_leaf(:,subgraph(i,
            subgraph(i,:) > 0));
181     odd_degree = find(mod(sum(G_sub,2),2) & sum(G_sub,2)>1);
182     odd_degree = reshape(odd_degree,length(odd_degree)/2,2);
183     for p = 1:size(odd_degree,1)
184         G_sub(odd_degree(p,1),odd_degree(p,2)) = 0;
185         G_sub(odd_degree(p,2),odd_degree(p,1)) = 0;
186     end
187     [row,col] = find(triu(G_sub));
188     sub_edge_list = [row col];
189     h = graph(sub_edge_list);
190     strategy = euler_trail(h);
191     %plug in previously deleted edges
192     for o = 1:length(odd_degree)/2
193         temp = find(strategy(:,2) == odd_degree(1));
194         strategy = [strategy(1:temp,:); odd_degree; strategy(temp + 1:
                end,:)];
195     end
196     %eulerian tour of the current connected leaf induced subgraph
197     u = strategy(1);
198     %insert an Edl edge at the start of Eulerian tour
199     v = find(G_edl(u,:)>0);
200     v = v(1);
```

```
201     edl_edge = setdiff(edl_edge,[u v],'rows');
202     edl_edge = setdiff(edl_edge,[v u],'rows');
203     G_edl(u,v) = 0;
204     G_edl(v,u) = 0;
205     % adjust tour of Ell edges to be the same form as Edl and Edd
206     strategy = fliplr(strategy);
207     strategy = [[u v]; strategy];
208     edge_list = [edge_list; strategy];
209     G_sub = zeros(max(max(original_graph)),max(max(original_graph)));
210 end
211
212 for i = 1:size(edl_edge,1)
213     if any(ismember(edl_edge(i,1),unique(cds_edge)))
214         edl_edge(i,:) = fliplr(edl_edge(i,:));
215     end
216 end
217
218 %child node cannot be the same, interleaving
219 [~,ix] = sort(edl_edge(:,2));
220 edl_edge = edl_edge(ix,:);
221 [~,ix] = sort(edl_edge(:,2));
222 ix = reshape(ix,length(ix)/2,2);
223 ix = ix';
224 ix = reshape(ix,size(ix,2)*2,1);
225 edl_edge = edl_edge(ix,:);
226 edge_list = [edge_list; edl_edge];
227 end
228
229 %identify the connected subgraph containing designated vertex
230 function augment = CommIdentify(graph,vertex)
231     augment = vertex;
```

```
232     previous_size = 0;
233     current_size = 1;
234     while current_size > previous_size
235         previous_size = length(augment);
236         l = length(augment);
237         for j = 1:l
238             neigh_sub = find(graph(augment(j),:)>0);
239             augment = [augment neigh_sub];
240         end
241         augment = unique(augment);
242         current_size = length(augment);
243     end
244 end
```

B.2.3 Function Transferring Edge Sequences to Protocols

```
1  function protocol = ProtocolDecompose(edge_list,Td_edge,none_leaf,tree)
2  % build a tree using edge_list, enabling using of matgraph
3  g = graph(tree);
4
5  % reshape edge list to find the number which appears more than once
6  % in edge_list. If a number appears more than once, that means it is
7  % not a leaf
8
9  % terminals of CDS
10 edge = reshape(Td_edge,2*size(Td_edge,1),1);
11 [n,bin] = histc(edge,unique(edge));
12 multiple = find(n==1);
13 index = ismember(bin,multiple);
14 terminal = unique(edge(index));
15 protocol = [];
```

```
16
17 for i = 1:size(edge_list,1)
18     %current initial dissemination node
19     I = edge_list(i,1);
20     %current estimate
21     estimate = sort(edge_list(i,:));
22     %next passive dissemination node, if next passive dissemination
           node is
23     %a terminal in the CDS, then the dissemination order should put the
24     %passive dissemination node to the last one disseminates
25     next = edge_list(mod(i,size(edge_list,1))+1,2);
26     if ismember(next,terminal)
27         %if current initial dissemination node is on CDS, then the
28         %dissemination length is the size of CDS
29         if ismember(I,none_leaf)
30             d = zeros(length(none_leaf) - 1,1);
31             % flexible CDS should exclude next passive dissemination
                   node
32             bone = setdiff(none_leaf,next);
33             bone = reshape(bone,length(bone),1);
34             for j = 1:length(bone)
35                 d(j) = dist(g,bone(j),I);
36             end
37             [~,IX] = sort(d);
38             bone = bone(IX);
39             temp = [bone estimate(1)*ones(length(bone),1)...
40             estimate(2)*ones(length(bone),1)];
41             protocol = [protocol; temp; [next estimate]];
42             % if it is a Edl type estimate, the flexibility is
                   decreased by
43             % one
```

```
44          elseif ismember(edge_list(i,2),none_leaf)
45                  second = edge_list(i,2);
46                  d = zeros(length(none_leaf) - 2,1);
47                  bone = setdiff(none_leaf,[next second]);
48                  bone = reshape(bone,length(bone),1);
49                  for j = 1:length(bone)
50                      d(j) = dist(g,bone(j),I);
51                  end
52                  [~,IX] = sort(d);
53                  bone = bone(IX);
54                  temp = [bone estimate(1)*ones(length(bone),1) estimate
                        (2)*ones(length(bone),1)];
55                  protocol = [protocol; I estimate; second estimate; temp
                        ;...
56                      [next estimate]];
57          else
58              d = zeros(length(none_leaf) - 1,1);
59              % flexible CDS should exclude next passive dissemination
                    node
60              bone = setdiff(none_leaf,next);
61              bone = reshape(bone,length(bone),1);
62              for j = 1:length(bone)
63                  d(j) = dist(g,bone(j),I);
64              end
65              [~,IX] = sort(d);
66              bone = bone(IX);
67              temp = [bone estimate(1)*ones(length(bone),1)...
68              estimate(2)*ones(length(bone),1)];
69              protocol = [protocol; I estimate; temp; [next estimate]];
70          end
71      else
```

```
72          if ismember(I,none_leaf)
73              d = zeros(length(none_leaf),1);
74              % flexible CDS should exclude next passive dissemination
                    node
75              bone = none_leaf;
76              bone = reshape(bone,length(bone),1);
77              for j = 1:length(bone)
78                  d(j) = dist(g,bone(j),I);
79              end
80              [~,IX] = sort(d);
81              bone = bone(IX);
82              temp = [bone estimate(1)*ones(length(bone),1)...
83              estimate(2)*ones(length(bone),1)];
84              protocol = [protocol; temp];
85          elseif ismember(edge_list(i,2),none_leaf)
86              second = edge_list(i,2);
87              d = zeros(length(none_leaf) - 1,1);
88              bone = setdiff(none_leaf,second);
89              bone = reshape(bone,length(bone),1);
90              for j = 1:length(bone)
91                  d(j) = dist(g,bone(j),I);
92              end
93              [~,IX] = sort(d);
94              bone = bone(IX);
95              temp = [bone estimate(1)*ones(length(bone),1)...
96              estimate(2)*ones(length(bone),1)];
97              protocol = [protocol; I estimate; second estimate; temp];
98          else
99              d = zeros(length(none_leaf),1);
100             % flexible CDS should exclude next passive dissemination
                    node
```

```
101                 bone = none_leaf;
102                 bone = reshape(bone,length(bone),1);
103                 for j = 1:length(bone)
104                     d(j) = dist(g,bone(j),I);
105                 end
106                 [~,IX] = sort(d);
107                 bone = bone(IX);
108                 temp = [bone estimate(1)*ones(length(bone),1)...
109                 estimate(2)*ones(length(bone),1)];
110                 protocol = [protocol; I estimate; temp];
111             end
112         end
113 end
114
115 end
```

Bibliography

[1] J. Proakis, *Digital Communications*, 4th ed. New York: McGraw-Hill, 2000.

[2] G. J. Foschini, "Layered space-time architecture for wireless communication in a fading environment when using multi-element antennas," *Bell Labs Tech. J.*, vol. 1, no. 2, pp. 41–59, 1996.

[3] S. Alamouti, "A simple transmit diversity technique for wireless communications," *Selected Areas in Communications, IEEE Journal on*, vol. 16, no. 8, pp. 1451 –1458, oct 1998.

[4] P. Chow, J. Cioffi, and J. Bingham, "A practical discrete multitone transceiver loading algorithm for data transmission over spectrally shaped channels," *Communications, IEEE Transactions on*, vol. 43, no. 234, pp. 773 –775, feb/mar/apr 1995.

[5] A. Scaglione, P. Stoica, S. Barbarossa, G. Giannakis, and H. Sampath, "Optimal designs for space-time linear precoders and decoders," *Signal Processing, IEEE Transactions on*, vol. 50, no. 5, pp. 1051 –1064, may 2002.

[6] B. Vojcic and W. M. Jang, "Transmitter precoding in synchronous multiuser communications," *Communications, IEEE Transactions on*, vol. 46, no. 10, pp. 1346 –1355, oct 1998.

[7] D. Love, J. Heath, R.W., W. Santipach, and M. Honig, "What is the value of limited feedback for mimo channels?" *Communications Magazine, IEEE*, vol. 42, no. 10, pp. 54 – 59, oct. 2004.

[8] J. Laneman, D. Tse, and G. Wornell, "Cooperative diversity in wireless networks: Efficient protocols and outage behavior," *Information Theory, IEEE Transactions on*, vol. 50, no. 12, pp. 3062 – 3080, dec. 2004.

[9] A. Sendonaris, E. Erkip, and B. Aazhang, "User cooperation diversity. part i. system description," *Communications, IEEE Transactions on*, vol. 51, no. 11, pp. 1927 – 1938, nov. 2003.

[10] P. Marsch and G. Fettweis, Eds., *Coordinated Multi-Point in Mobile Communications - From Theory to Practice*. Cambridge University Press, 2011.

[11] M. Maddah-Ali, A. Motahari, and A. Khandani, "Communication over mimo x channels: Interference alignment, decomposition, and performance analysis," *Information Theory, IEEE Transactions on*, vol. 54, no. 8, pp. 3457 –3470, aug. 2008.

[12] A. Sendonaris, E. Erkip, and B. Aazhang, "Interference alignment and degrees of freedom of the k-user interference channel," *Information Theory, IEEE Transactions on*, vol. 54, no. 8, pp. 3425 – 3441, Aug. 2008.

[13] G. Kramer, M. Gastpar, and P. Gupta, "Cooperative strategies and capacity theorems for relay networks," *Information Theory, IEEE Transactions on*, vol. 51, no. 9, pp. 3037 – 3063, sept. 2005.

[14] R. Mudumbai, G. Barriac, and U. Madhow, "On the feasibility of distributed beamforming in wireless networks," *Wireless Communications, IEEE Transactions on*, vol. 6, no. 5, pp. 1754 –1763, may 2007.

[15] L. Zhang, Y.-C. Liang, Y. Xin, and H. Poor, "Robust cognitive beamforming with partial channel state information," *Wireless Communications, IEEE Transactions on*, vol. 8, no. 8, pp. 4143 –4153, august 2009.

[16] A. Ibrahim, A. Sadek, W. Su, and K. Liu, "Cooperative communications with partial channel state information: When to cooperate?" in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 5, dec. 2005, pp. 5 pp. –3072.

[17] S. Ramprashad and G. Caire, "Cellular vs. network mimo: A comparison including the channel state information overhead," in *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, sept. 2009, pp. 878 –884.

[18] B. Chalise and L. Vandendorpe, "Mimo relay design for multipoint-to-multipoint communications with imperfect channel state information," *Signal Processing, IEEE Transactions on*, vol. 57, no. 7, pp. 2785 –2796, july 2009.

[19] S. Jafar, "Exploiting channel correlations - simple interference alignment schemes with no csit," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, dec. 2010, pp. 1 –5.

[20] W.-J. Huang, Y.-W. Hong, and C.-C. Kuo, "Lifetime maximization for amplify-and-forward cooperative networks," *Wireless Communications, IEEE Transactions on*, vol. 7, no. 5, pp. 1800 –1805, may 2008.

[21] Y.-W. Hong, W.-J. Huang, F.-H. Chiu, and C.-C. Kuo, "Cooperative communications in resource-constrained wireless networks," *Signal Processing Magazine, IEEE*, vol. 24, no. 3, pp. 47 –57, may 2007.

[22] R. Mudumbai, D. Brown, U. Madhow, and H. Poor, "Distributed transmit beamforming: challenges and recent progress," *Communications Magazine, IEEE*, vol. 47, no. 2, pp. 102 –110, february 2009.

[23] D. Lopez-Perez, A. Valcarce, G. de la Roche, and J. Zhang, "Ofdma femtocells: A roadmap on interference avoidance," *Communications Magazine, IEEE*, vol. 47, no. 9, pp. 41 –48, september 2009.

[24] A. Ozgur, O. Leveque, and D. Tse, "Hierarchical cooperation achieves optimal capacity scaling in ad hoc networks," *Information Theory, IEEE Transactions on*, vol. 53, no. 10, pp. 3549 –3572, oct. 2007.

[25] S. Ramanan and J.-L. Walsh, "Distributed estimation of channel gains in wireless sensor networks," *Signal Processing, IEEE Transactions on*, vol. 58, no. 6, pp. 3097 –3107, june 2010.

[26] C. Esli and A. Wittneben, "A cluster-based multiuser cooperative network," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, nov. 2007, pp. 3926 –3931.

[27] G. Khanna, S. Bagchi, and Y.-S. Wu, "Fault tolerant energy aware data dissemination protocol in sensor networks," in *Dependable Systems and Networks, 2004 International Conference on*, june-1 july 2004, pp. 795 – 804.

[28] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: design, analysis and applications," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3, march 2005, pp. 1653 – 1664 vol. 3.

[29] T. M. Cover and T. Joy A., "Elements of information theory," *Hoboken, NJ: Wiley-Interscience. Print*, 2006.

[30] J. Wu, M. Gao, and I. Stojmenovic, "On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks," in *Parallel Processing, International Conference on, 2001.* IEEE, 2001, pp. 346–354.

[31] B. Das and V. Bharghavan, "Routing in ad-hoc networks using minimum connected dominating sets," in *Communications, 1997. ICC 97 Montreal,'Towards the Knowledge Millennium'. 1997 IEEE International Conference on*, vol. 1. IEEE, 1997, pp. 376–380.

[32] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 13, no. 1, pp. 14–25, 2002.

[33] J. M. Harris, H. J. L., and M. M. J., "Combinatorics and graph theory," *Springer Verlag. Print*, 2008.

[34] R. Anitha and R. Lekshmi, "N-sun decomposition of complete, complete bipartite and some harary graphs," *International Journal of Computational and Mathematical Sciences*, vol. 2, pp. 33–38, 2008.

[35] E. Lucas, "K ecrkeations mathkematiques, t ome ii," *Albert Blanchard, Paris*, 1892 (reprinted 1975).

[36] E. J. Wegman, "Hyperdimensional data analysis using parallel coordinates," *Journal of the American Statistical Association*, vol. 85, no. 411, pp. 664–675, Sep. 1990.

[37] P.-J. Wan, K. M. Alzoubi, and O. Frieder, "Distributed construction of connected dominating set in wireless ad hoc networks," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3. IEEE, 2002, pp. 1597–1604.

[38] S. Guha and S. Khuller, "Approximation algorithms for connected dominating sets," *Algorithmica*, vol. 20, no. 4, pp. 374–387, 1998.

[39] J. R. Griggs, D. J. Kleitman, and A. Shastri, "Spanning trees with many leaves in cubic graphs," *Journal of Graph Theory*, vol. 13, no. 6, pp. 669–695, 1989.

[40] D. J. Kleitman and D. B. West, "Spanning trees with many leaves," *SIAM Journal on Discrete Mathematics*, vol. 4, no. 1, pp. 99–106, 1991.

[41] G. Wen-Yu, "Kernelization algorithm of maximum leaf spanning tree," *Chinese Journal of Computers*, vol. 33, no. 12, 2010.