

OVERLAY NETWORK CONSTRUCTION IN HIGHLY DECENTRALIZED
NETWORKS

by

Melih Onus

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

ARIZONA STATE UNIVERSITY

May 2009

OVERLAY NETWORK CONSTRUCTION IN HIGHLY DECENTRALIZED
NETWORKS

by

Melih Onus

has been approved

April 2009

Graduate Supervisory Committee:

Andréa W. Richa, Chair
Goran Konjevod
Christian Scheideler
Rida Bazzi

ACCEPTED BY THE GRADUATE COLLEGE

ABSTRACT

In recent years, highly decentralized networks such as wireless ad hoc and peer-to-peer (P2P) have emerged as the most interesting, challenging and innovation rich areas in computer networking. The nature of these networks require efficient, local, scalable and self-stabilizing algorithmic solutions.

An important problem in highly decentralized networks is constructing an overlay network on top of which one can efficiently provide services such as routing, broadcasting and information gathering in wireless ad hoc networks and routing, data search and redundant storage in P2P networks.

In this thesis, the problem of overlay network construction in highly decentralized networks is studied. Firstly, our work on overlay network construction for wireless ad-hoc networks is presented. A more general communication model for wireless ad-hoc networks, and a locally self-stabilizing algorithm to construct an overlay network is presented. More precisely, a backbone structure is established by efficiently electing cluster leaders and gateway nodes. For electing cluster leaders, Luby's maximal independent set algorithm is extended to our more complex model and then every pair of cluster leaders which are close to each other is interconnected with the help of gateway nodes. Secondly, our work on construction of line-based P2P overlay networks is presented. Locally self stabilizing algorithms for constructing peer-to-peer overlay networks is presented. Linearization is proposed as the basic technique to sort any connected graph. The linearization technique is self-stabilizing and transforms any connected graph into a sorted list. Lastly our work on construction of publish-subscribe P2P overlay networks is presented. An overlay network is designed for publish/subscribe communication in a system where nodes may subscribe

to many different topics of interest. The following problem is considered: Given a collection of nodes and their topic subscriptions connect the nodes into a graph which has least possible maximum degree and in such a way that for each topic t , the graph induced by the nodes interested in t is connected. The first polynomial time logarithmic approximation algorithm is presented for this problem and an almost tight lower bound on the approximation ratio is presented. Our algorithm is based on greedy matching algorithm for graphs. Experimental results show that our algorithm drastically improves the maximum degree of publish/subscribe overlay systems.

Dedicated to my family and her

TABLE OF CONTENTS

	Page
CHAPTER 1 INTRODUCTION	1
LIST OF TABLES	1
LIST OF FIGURES	1
CHAPTER 2 OVERLAY NETWORK CONSTRUCTION FOR WIRELESS AD- HOC NETWORKS	5
2.1. Wireless communication model	6
2.1.1. Carrier sensing	7
2.1.2. Transmission range, interference range, and physical carrier sens- ing range	8
2.1.3. Our model	10
2.2. Constant Density Spanners for Wireless Ad-Hoc Networks	13
2.2.1. Our contributions	13
2.2.2. Previous work	16
2.2.3. Overview of Spanner Protocol	18
2.2.4. Phase I: Dominating Set	20
2.2.5. Constant Density Spanner	30
2.3. Broadcasting and Information Gathering in Wireless Ad-Hoc Networks . . .	38
2.3.1. Isolated Broadcasting	40
2.3.2. Broadcasting Multiple Messages	44
2.3.3. Information Gathering	49

	Page
2.4. Conclusion	58
CHAPTER 3 CONSTRUCTION OF LINE-BASED P2P OVERLAY NETWORKS	60
3.1. Our contributions	64
3.2. Pure Linearization	65
3.2.1. Analysis.	67
3.2.2. Other Properties of PL	72
3.3. Linearization with Memory	74
3.4. Linearization with Shortcut Neighbors	78
3.5. Experimental Results	80
3.5.1. Regular Graphs	80
3.5.2. Power Law Graphs	82
3.5.3. Bounded Locality	83
3.6. Conclusion	84
CHAPTER 4 PUBLISH-SUBSCRIBE OVERLAY NETWORK DESIGN	85
4.1. Related Work	89
4.2. Minimum Maximum Degree Publish-Subscribe Overlay Network Design .	91
4.2.1. Preliminaries	91
4.2.2. MinAv-TCO problem and Greedy Merge(GM) Algorithm	92
4.2.3. Overlay Design Algorithm	94
4.2.4. Approximation Ratio	96
4.2.5. Hardness of MinMax-TCO Problem	102

	Page
4.2.6. Experimental Results	104
4.3. Parameterized Maximum and Average Degree Approximation In Topic- Based Publish-Subscribe Overlay Network Design	108
4.3.1. Minimizing The Maximum Degree Only	108
4.3.2. Low Degree Overlay Design Algorithm (Low-ODA)	109
4.3.3. Approximation Ratio	112
4.4. Constructing Constant Diameter Overlays for Publish-Subscribe	120
4.4.1. The GM Algorithm and the CD-TCO Problem	121
4.4.2. Examples I and II	122
4.4.3. Constant Diameter Overlay Design Algorithm (CD-ODA)	123
4.4.4. Example III	123
4.4.5. Constant Diameter Overlay Design Algorithm I(CD-ODA-I)	125
4.4.6. Example IV	125
4.4.7. Constant Diameter Overlay Design Algorithm II(CD-ODA-II)	126
4.4.8. Analysis of Algorithms	127
4.4.9. Conjectures	128
4.4.10. Experimental Results	128
4.5. Conclusion	131
REFERENCES	133

LIST OF TABLES

Table		Page
1.	Average number of rounds for PL, LM and LSN (for each node avg number of neighbors is 10)	81
2.	Average maximum node degree for LM and LSN (for each node avg number of neighbors is 10)	81
3.	Average number of rounds for PL, LM and LSN (graphs satisfy power law with $\alpha = 2$)	83
4.	Average number of rounds for PL, LM and LSN with bounded locality . . .	83
5.	Summary of known results on overlay network construction for publish/subscribe communication (n : number of nodes, t : number of topics, c : constant, k is any parameter between 1 and n)	88

LIST OF FIGURES

Figure		Page
1.	Figure (a) shows the notion of transmission range in terms of cost of communication. Notice that node u can communicate directly with nodes v and a and c but not with nodes b or w . Figures (b) shows the sensing ranges. When all nodes use a transmission power P and node w uses a threshold of T , in this example, node w can <i>always</i> sense transmissions of node a while it <i>may</i> sense transmissions of node b and <i>can never</i> sense transmissions of node c	12
2.	Two consecutive rounds of the spanner protocol.	20
3.	The spanner of the original network.	21
4.	Left and right neighbors of node u	66
5.	Neighbors of node u after one round of pure linearization.	67
6.	Lower bound example for PL algorithm	68
7.	Node u places its current left neighbors into sorted list by reconnecting them.	75
8.	Intervals for node v	79
9.	Average number of rounds for PL, LM and LSN	81
10.	Average maximum node degree for LM and LSN (for each node avg number of neighbors is 10)	82
11.	Average number of rounds for PL, LM and LSN (graphs satisfy power law with $\alpha = 2$)	83
12.	Average number of rounds for PL, LM and LSN with bounded locality	84

Figure	Page
13. (a) Overlay with optimal max degree (b) Overlay constructed by GM	93
14. Maximum node degree for GM and MinMax-ODA (a) Number of topics is 100 (b) Number of topics is 200	105
15. Average node degree for GM and MinMax-ODA (a) Number of topics is 100 (b) Number of topics is 200	106
16. Average and maximum node degree for different subscription size (Number of nodes and number of topics is 100)	107
17. (a) Example I (b) Example II	122
18. (a) CD-ODA on Example III (b) Optimal solution for Example III	124
19. (a) CD-ODA and CD-ODA-I on Example IV (b) Optimal solution for Ex- ample IV	126
20. Average node degree for GM, CD-ODA, CD-ODA-I and CD-ODA-II . . .	129
21. Average node degree for GM, CD-ODA, CD-ODA-I and CD-ODA-II . . .	131

CHAPTER 1

INTRODUCTION

In recent years, highly decentralized networks such as wireless ad hoc and peer-to-peer (P2P) have emerged as the most interesting, challenging and innovation rich areas in computer networking. The nature of these networks require efficient, local, scalable and self-stabilizing algorithmic solutions.

An important problem in highly decentralized networks is constructing an overlay network on top of which one can efficiently provide services such as routing, broadcasting and information gathering in wireless ad hoc networks and routing, data search and redundant storage in P2P networks.

In this thesis, the problem of overlay network construction in highly decentralized networks is studied. This thesis has three parts. In the first part, our work on overlay network construction for wireless ad-hoc networks is presented. In the second part, our work on construction of line-based P2P overlay networks is presented. In the third part, our work on construction of publish-subscribe P2P overlay networks is presented.

In Chapter 2, our work on overlay network construction for wireless ad-hoc networks is presented [Kothapalli et al., 2005, Onus et al., 2005]. An important problem for wireless ad hoc networks has been to design overlay networks that allow time- and energy-efficient routing. Many local-control strategies for maintaining such overlay networks have already been suggested, but most of them are based on an oversimplified wireless communication model.

In this thesis, we suggest a model that is much more general than previous models. It allows the path loss of transmissions to significantly deviate from the idealistic unit disk

model and does not even require the path loss to form a metric. Also, our model is apparently the first proposed for algorithm design that does not only model transmission and interference issues but also aims at providing a realistic model for physical carrier sensing. Physical carrier sensing is needed so that our protocols do not require any prior information (not even an estimate on the number of nodes) about the wireless network to run efficiently.

Based on this model, we propose a local-control protocol for establishing a constant density spanner among a set of mobile stations (or nodes) that are distributed in an arbitrary way in a 2-dimensional Euclidean space. More precisely, we establish a backbone structure by efficiently electing cluster leaders and gateway nodes so that there is only a constant number of cluster leaders and gateway nodes within the transmission range of any node and the backbone structure satisfies the properties of a topological spanner.

Our protocol has the advantage that it is locally self-stabilizing, i.e., it can recover from any initial configuration, even if adversarial nodes participate in it, as long as the honest nodes sufficiently far away from adversarial nodes can in principle form a single connected component. Furthermore, we only need constant size messages and a constant amount of storage at the nodes, irrespective of the distribution of the nodes. Hence, our protocols would even work in extreme situations such as very simple wireless devices (like sensors) in a hostile environment. We also designed broadcasting and information gathering algorithms for wireless ad hoc networks.

Secondly, in Chapter 3, our work on construction of line-based P2P overlay networks is presented [Onus et al., 2007]. We consider the problem of designing a distributed algorithm that, given an arbitrary connected graph G of nodes with unique labels, converts

G into a sorted list of nodes. This algorithm should be as simple as possible and, for scalability, should guarantee a polylogarithmic runtime as well as at most a polylogarithmic increase in the degree of each node during its execution. Furthermore, it should be self-stabilizing, that is, it should be able to eventually construct a sorted list from any state in which the graph is connected. It turns out that satisfying all of these demands at the same time is not easy.

Our basic approach towards this goal is the so-called linearization technique: each node v repeatedly does the following with its neighbors:

- for its left (i.e., smaller) neighbors $u_1, \dots, u_k$ in the order of decreasing labels, v replaces $\{v, u_1\}, \dots, \{v, u_k\}$ by $\{v, u_1\}, \{u_1, u_2\}, \dots, \{u_{k-1}, u_k\}$, and
- for its right (i.e., larger) neighbors $w_1, \dots, w_\ell$ in the order of increasing labels, v replaces $\{v, w_1\}, \dots, \{v, w_\ell\}$ by $\{v, w_1\}, \{w_1, w_2\}, \dots, \{w_{\ell-1}, w_\ell\}$.

As shown in this thesis, this technique transforms any connected graph into a sorted list, but there are graphs for which this can take a long time. Hence, we propose several extensions of the linearization technique and experimentally evaluate their performance. Our results indicate that some of these have a polylogarithmic performance, so there is hope that there are distributed algorithms that can achieve all of our goals above.

Lastly, in Chapter 4, our work on construction of publish-subscribe P2P overlay networks is presented [Onus and Richa, 2009b, Onus and Richa, 2009a]. Designing an overlay network for publish/subscribe communication in a system where nodes may subscribe to many different topics of interest is of fundamental importance. For scalability and efficiency, it is important to keep the degree of the nodes in the publish/subscribe system low.

It is only natural then to formalize the following problem: Given a collection of nodes and their topic subscriptions connect the nodes into a graph which has least possible maximum degree and in such a way that for each topic t , the graph induced by the nodes interested in t is connected. We present the first polynomial time logarithmic approximation algorithm for this problem and prove an almost tight lower bound on the approximation ratio. Our experimental results show that our algorithm drastically improves the maximum degree of publish/subscribe overlay systems. Furthermore, we presented an algorithm which approximates both average degree and maximum degree well.

We also propose a variation of the problem by enforcing that each topic-connected overlay network be of constant diameter, while keeping the average degree low. We present three heuristics for this problem which guarantees that each topic-connected overlay network will be of diameter 2 and which aims at keeping the overall average node degree low. Our experimental results validate our algorithms showing that our algorithm is able to achieve very low diameter without increasing the average degree by much.

CHAPTER 2

OVERLAY NETWORK CONSTRUCTION FOR WIRELESS AD-HOC NETWORKS

An important problem for wireless ad hoc networks has been to design overlay networks that allow time- and energy-efficient routing. Many local-control strategies for maintaining such overlay networks have already been suggested, but mostly high-level wireless models have been used for their analysis. However, since mobile ad-hoc networks have many features that are hard to model in a clean way, it is not clear how well these strategies may actually perform in practice. Major challenges are how to model wireless communication and how to model mobility. Here, theoretical work is still in its infancy. So far, people in the algorithms community have mostly looked at static wireless networks (i.e. the wireless stations are always available and do not move). Wireless communication has mostly been modeled using the packet radio network or the even simpler unit disk model. In the packet radio network model, the wireless units, or nodes, are represented by a graph, and two nodes are connected by an edge if and only if they are within transmission range of each other. Transmissions of messages interfere at a node if at least two of its neighbors transmit a message at the same time. A node can only receive a message if it does not interfere with any other message.

The packet radio network model is a simple and clean model that allows to design and analyze algorithms for overlay networks with a reasonable amount of effort. However, since it is a high-level model, it does have some serious problems with certain scenarios in practice because in reality the transmission range of a message is not the same as its interference range. Consider, for example, two nodes s and t and a set U of n nodes with

all nodes in U being within the transmission range of s but only one node in U , v , being within the transmission range of t . All other nodes in U just interfere with t . Then no uniform protocol (i.e., at each step all nodes try to access the wireless medium with the same probability) can send a message from s to t in an expected $o(n)$ number of steps, whereas a constant expected time is easy to achieve if we can ignore interference problems between U and t .

There are a limited number of papers in the theory area that use a model that differentiates between the transmission range and the interference range [Adler and Scheideler, 2000, Gupta and Kumar, 2000, Gupta and Kumar, 2001a, Gupta and Kumar, 2001b, Kuhn et al., 2003b], but they still assume a disk model in a sense that the transmission range and interference range can be modeled by two distance values that hold irrespective of the position of a node. We propose a more general model.

2.1. Wireless communication model

In order to motivate our model, we first review some commonly used transmission techniques in wireless communication. We will concentrate here on the IEEE 802.11 standard because IEEE 802.11-based radio LANs are currently dominating the market and will most probably do so also in the future. The IEEE 802.11 standard distinguishes between a Physical (PHY) layer and a Medium Access Control (MAC) layer for the transmission of messages. The 802.11 MAC protocols are based on Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA).

2.1.1. Carrier sensing

The basic approach of the CSMA/CA scheme is as follows. Whenever a node receives a packet to be transmitted, it first listens to the channel to ensure no other node is transmitting. If the channel is clear, it transmits the packet. Otherwise, it uses an exponential back-off scheme until it either finds a time point in which the channel is clear so that it can transmit its packet or aborts the transmission due to too many failed attempts.

In WLAN devices, there is usually just one antenna for both sending and receiving, and hence the stations are not able to listen while sending. For this and other reasons there is no collision detection capability like in the Ethernet. Therefore, acknowledgment packets (ACK) have to be sent from the receiver to the sender to confirm that packets have been correctly received.

In wireless ad hoc networks that rely on a carrier-sensing random access protocol, such as IEEE 802.11, the wireless medium characteristics generate complex phenomena such as the well-known hidden-station and exposed-station problems. In order to handle these problems, the MAC layer uses physical and virtual carrier sensing techniques.

The physical carrier sensing part of the CSMA scheme is provided by a Clear Channel Assessment (CCA) circuit. This circuit monitors the environment to determine when it is clear to transmit. It can be programmed to be a function of the Receive Signal Strength Indication (RSSI) and other parameters. The RSSI measurement is derived from the state of the Automatic Gain Control (AGC) circuit. Whenever the RSSI exceeds a certain threshold, a special Energy Detection (ED) bit is switched to 1, and otherwise it is set to 0. By manipulating a certain configuration register, this threshold may be set to an absolute power

value of t dB, or it may be set to be t dB above the measured noise floor, where t can be set to any value in the range 0-127. The ability to manipulate the CCA rule allows the MAC layer to optimize the physical carrier sensing to its needs.

Virtual carrier sensing is usually achieved by using two control packets, Request To Send (RTS) and Clear To Send (CTS), which are exchanged before the data transmission is taking place. More precisely, before transmitting a data frame, the source station sends an RTS packet to the receiving station announcing the upcoming frame transmission. Upon receiving the RTS packet, the destination replies by a CTS packet to indicate that it is ready to receive the data frame. Both the RTS and CTS packets contain the total duration of the transmission, i.e. the overall time needed to transmit the data frame and the related ACK, so that other stations within the transmission range of either the source or the destination stay silent until the transmission is over.

2.1.2. Transmission range, interference range, and physical carrier sensing range

Every data transmission mechanism has a minimum signal-to-noise ratio (SNR) at which a data frame can still be transmitted with a reasonably low frame error rate. The minimum SNRs for 802.11b, for example, are 10dB for 11Mbps, 8dB for 5.5Mbps, 6dB for 2Mbps, and 4dB for 1Mbps, and for 802.11a, 23dB is usually the minimum SNR for 54Mbps. In the 802.11a standard [P802.11a/D7.0-1999, 1999], the minimum dB values are defined as the received signal strength level at which the frame error rate (FER) of a 1000-octet frame is less than 10

The SNRs above specify the transmission range of the data transmission mech-

anism, i.e. the maximum range within which data frames can still be received correctly. The transmission range is highly dependent on the environment. A reasonable model for determining the transmission range is the log-normal shadowing model [Kuruwila et al., 2004, Quin and Kunz, 2003]. In this model, the received power at a distance of d relative to the received power at a reference distance of d_0 is given in dB as, $-10 \log(d/d_0)^\theta + X_\sigma$, where θ is the path loss coefficient and X_σ is a Gaussian random variable with zero mean and standard deviation σ (in dB) that models the influence of the background noise. θ usually ranges from 2 (free space) to 5 (indoors) [Rappaport, 1996].

When using forward error correction mechanisms as proposed in the IEEE 802.11e MAC standard currently under development, the transition between being able to correctly receive a data frame with high probability and not being able to correctly receive a data frame with high probability is very sharp. As shown in [Choi, 2002], it can be less than 1 dB. Thus, in an ideal environment the transmission range is an area with a relatively sharp border that in reality, however, may be blurred due to environmental effects.

A limitation of the shadowing model is that it is only applicable in uniform environments. In non-uniform environments, the signal strength can exhibit a non-monotonic behavior. For example, it can happen that the sender position A has a smaller distance to a position B than to a position C and yet the strength of the signal from A received at B is lower than the signal strength received at C . This can even happen if B and C are close by.

For the interference and physical carrier sensing ranges there does not seem to be a commonly accepted definition in practice. So we will use a conservative model for these ranges to make sure that our results in this model are meaningful in practice.

2.1.3. Our model

In our model, we assume that we are given a set V of mobile stations, or *nodes*, that are distributed in an arbitrary way in a 2- dimensional Euclidean space. For any two nodes $v, w \in V$ let $d(v, w)$ be the Euclidean distance between v and w . Furthermore, consider any cost function c with the property that there is a fixed constant $\delta \in [0, 1)$ so that for all $v, w \in V$,

$$c(v, w) \in [(1 - \delta) \cdot d(v, w), (1 + \delta) \cdot d(v, w)] \text{ and } c(v, w) = c(w, v), \text{ i.e. } c \text{ is}$$

symmetric.

c determines the transmission and interference behavior of nodes and δ bounds the non-uniformity of the environment. Notice that we do not require c to be monotonic in the distance or to satisfy the triangle inequality. This makes sure that our model even applies to highly irregular environments. In Figure 1(a), for example, the distance between u and v is greater than the distance between u and w . Yet, the cost of communicating between u and w , $c(u, w)$, is bigger than $c(u, v)$. Similar cost functions were also used in [Kuhn et al., 2003a].

We assume that the nodes use some fixed-rate power-controlled communication mechanism over a single frequency band. When using a transmission power of P , there is a transmission range $r_t(P)$ and an interference range $r_i(P) > r_t(P)$ that grow monotonically with P . The interference range has the property that every node $v \in V$ can only cause interference at nodes w with $c(v, w) \leq r_i(P)$, and the transmission range has the property that for every two nodes $v, w \in V$ with $c(v, w) \leq r_t(P)$, v is guaranteed to receive a message from w sent out with a power of P (with high probability) as long as there is no other

node $v' \in V$ with $c(v, v') \leq r_i(P')$ that transmits a message at the same time with a power of P' .

For simplicity, we assume that the ratio $\rho = r_i(P)/r_t(P)$ is a fixed constant greater than 1 for all relevant values of P . This is not a restriction because we do not assume anything about what happens if a message is sent from a node v to a node w within v 's transmission range but another node u is transmitting a message at the same time with w in its interference range. In this case, w may or may not be able to receive the message from v , so any worst case must be assumed in the analysis. The only restriction we need, which is important for any overlay network algorithm to eventually stabilize, is that the transmission range is a sharp threshold. That is, beyond the transmission range a message cannot be received any more (with high probability). This is justified by the fact that when using modern forward error correction techniques, the difference between the signal strength that allows to receive the message (with high probability) and the signal strength that does not allow any more to receive the message (with high probability) can be very small (less than 1 dB).

Nodes can not only send and receive messages but also perform physical carrier sensing, which has not been considered before in models proposed in the algorithms community. Given some sensing threshold T (that can be flexibly set by a node) and a transmission power P , there is a *carrier sense transmission (CST) range* $r_{st}(T, P)$ and a *carrier sense interference (CSI) range* $r_{si}(T, P)$ that grow monotonically with T and P . The range $r_{st}(T, P)$ has the property that if a node v transmits a message with power P and a node w with $c(v, w) \leq r_{st}(T, P)$ is currently sensing the carrier with threshold T , then w senses a

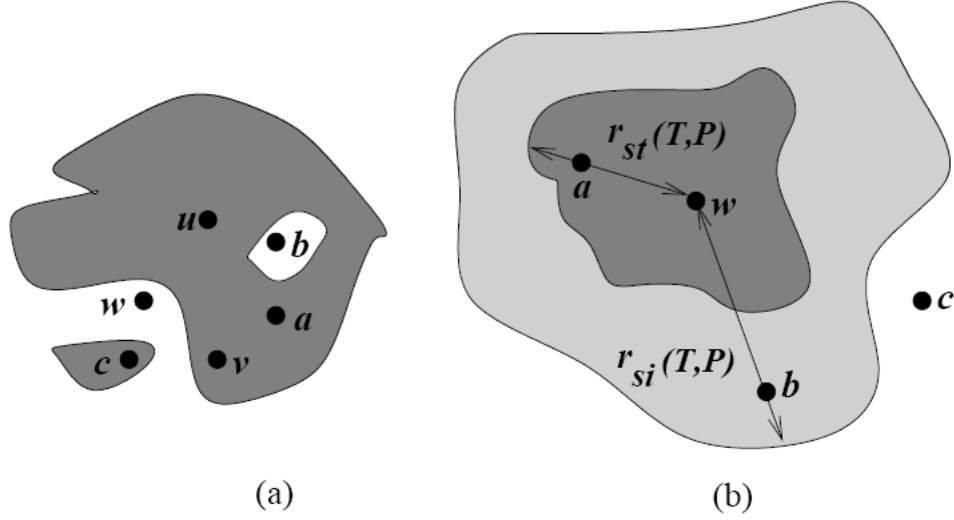


Figure 1. Figure (a) shows the notion of transmission range in terms of cost of communication. Notice that node u can communicate directly with nodes v and a and c but not with nodes b or w . Figure (b) shows the sensing ranges. When all nodes use a transmission power P and node w uses a threshold of T , in this example, node w can *always* sense transmissions of node a while it *may* sense transmissions of node b and *can never* sense transmissions of node c .

message transmission (with high probability). The range $r_{si}(T, P)$ has the property that if a node v senses a message transmission with threshold T , then there was at least one node w with $c(v, w) \leq r_{si}(T, P)$ that transmitted a message with power P (with high probability). More precisely, we assume that the monotonicity property holds. That is, if transmissions from a set U of nodes within the $r_{si}(T, P)$ range cause v to sense a transmission, then any superset of U will also do so. The two sensing ranges are shown in Figure 1.

For simplicity, we will assume in the following that for the carrier sense ranges, $r_{si}(T, P)/r_{st}(T, P) = r_i(P)/r_t(P)$ for all relevant values of T .

2.2. Constant Density Spanners for Wireless Ad-Hoc Networks

2.2.1. Our contributions

Our contributions are two-fold: we present a new model for wireless networks, and we demonstrate how to develop and analyze algorithms on top of this model by presenting self-stabilizing local-control algorithms for building constant density dominating sets and spanners.

In our algorithms, the nodes do not have to have any a priori knowledge about the other nodes, not even an estimate on their total number. Also, fixed identification numbers of any form are not required so that our protocols may even be applicable to the important field of sensor networks. It is sufficient for us if the nodes choose identification numbers so that there are no local conflicts (which can be easily achieved with random, local-control coloring strategies). We only require that the mobile hosts can synchronize in rounds of constant length. This can be done, for example, with the help of GPS signals or any form of beacons (that are sufficiently far apart in time for a round of our protocols to complete).

In order to obtain a constant density spanner under an arbitrary distribution of nodes, we proceed in two stages. First, we show that there is a simple, distributed protocol to obtain a constant density dominating set, and then we show how to extend this protocol in order to also obtain a constant density spanner.

It is worth noting that our protocols only need a constant amount of storage at each node, irrespective of the distribution of the nodes. The constant only depends on the δ in our model. Moreover, our protocols can self-stabilize even if some of the nodes show arbitrary adversarial behavior. We only require the honest nodes that are outside a certain

range of the adversarial nodes to be placed so that they can in principle form a single connected component. So our protocols would even work for very primitive devices in hostile environments.

2.2.1.1. CONSTANT DENSITY DOMINATING SET

Given an undirected graph $G = (V, E)$, a subset $U \subseteq V$ is called a dominating set if all nodes $v \in V$ are either in U or have an edge to a node in U . A dominating set U is called connected if U forms a connected component in G . The density of a dominating set is the maximum over all nodes $v \in U$ of the number of neighbors that v has in U .

Given an arbitrary distribution of a set V of nodes in a 2-dimensional Euclidean space, let the graph $Q_r = (V, E_r)$ contain all edges v, w with $d(v, w) \leq r$. Suppose that we select a maximal independent set U in Q_r . Then this is also a dominating set of constant density because in the 2-dimensional Euclidean space a node can have at most five neighbors within a distance of r that are part of an independent set in Q_r [Alzoubi et al., 2002b]. Note that a constant density dominating set is also a constant factor approximation of a minimum dominating set, a well-studied problem in the algorithms and wireless networking community.

Now, let us consider the graph $G_r = (V, E'_r)$ that contains all edges $\{v, w\}$ such that $c(v, w) \leq r$. Since $c(v, w) \leq (1 + \delta)d(v, w)$, it follows from [Alzoubi et al., 2002b]:

Fact 2.2.1 *Every node v can have at most five neighbors within a Euclidean distance of $r/(1 + \delta)$ that are part of an independent set in G_r .*

Otherwise, there must be a pair $v, w \in V$ with $c(v, w) \leq (1 + \delta)d(v, w) \leq (1 + \delta) \cdot$

$r/(1 + \delta) = r$ that are in an independent set in G_r , a contradiction. Furthermore, because $c(v, w) \geq (1 + \delta)d(v, w)$, a node can only be connected in G_r to nodes up to a Euclidean distance of $r/(1 - \delta)$. Hence, it is easy to see that for every node v there is a set C_v of neighbors of v in G_r of constant size so that for every neighbor w of v in G_r there is a neighbor $w' \in C_v$ with $d(w, w') \leq r/(1 + \delta)$. Combining this with Fact 2.2.1, we get:

Fact 2.2.2 *For any independent set in G_r it holds that every node v in G_r can have at most a constant number of neighbors in this set, where the constant depends on δ .*

Now, recall that any maximal independent set in a graph G_r is also a dominating set in G_r , and according to the fact above, any maximal independent set in G_r has a constant density (i.e., every node only has a constant number of neighbors in that set). Hence, in order to obtain a dominating set of constant density, it suffices to design an algorithm that constructs a maximal independent set in G_r . It turns out that constructing such a set is quite tricky, given the uncertainties in our model, but we can construct something close to that so that the following result holds.

Theorem 2.2.3 *For any desired transmission range r and any initial situation, the dominating set protocol generates a constant density dominating set in G_r in $O(\log^4 n)$ communication rounds, with high probability.*

Hence, our protocol self-stabilizes within $O(\log^4 n)$ rounds. Interestingly, this result is only possible because our protocol uses physical carrier sensing. It is known that if physical carrier sensing is not available and the nodes have no estimate of the size of the network, then it takes $\Omega(n)$ steps on expectation for a single message transmission to be successful [Jurdzinski and Stachowiak, 2002] in any protocol.

2.2.1.2. CONSTANT DENSITY SPANNER

A subgraph H of a graph G is called a (*topological*) t -*spanner* of G if for every pair of nodes $v, w \in G$ there is a path in H from v to w whose length is at most t times the minimum length of a path from v to w in G . In this case, t is also called the stretch factor of H .

We then extend the dominating set protocol by additional protocols that connect the nodes in the dominating set via so-called gateway nodes so that the following result holds.

Theorem 2.2.4 *For any desired transmission range r and any initial situation, the spanner protocol generates a constant density spanner in G_r in $O(D \log D \log n + \log^4 n)$ communication rounds, with high probability, where D is the maximum number of nodes that are within the transmission range of a node.*

All of our protocols can self-stabilize even under adversarial behavior as long as the nodes outside a range of $r' = \Theta(r)$ of adversarial nodes form a connected component in G_r .

2.2.2. Previous work

The problem of finding a minimum dominating set has been shown to be NP-complete even when restricted to unit disk graphs [Clark et al., 1990] and, hence, approximation algorithms are of interest. Recent research focused on developing distributed (rather than centralized) algorithms for finding good approximations of minimum dominating sets in arbitrary graphs (see, for example, [Dubhashi et al., 2003, Jia et al., 2001, Kuhn and Wattenhofer, 2003]). A simple and elegant distributed approximation algorithm

was proposed by Luby [Luby, 1985].

Alzoubi et al. [Alzoubi et al., 2002a] presented the first constant approximation algorithm for the minimum connected dominating set problem in unit disk graphs with $O(n)$ and $O(n \log n)$ time and message complexity, respectively. Cheng et al. [Cheng et al., 2003] proposed a polynomial time approximation scheme for the connected dominating set problem in unit-disk graphs.

Huang et al. [Huang et al., 2004] formally analyze a popular algorithm used for clustering in ad-hoc mobile network scenarios. They show that this algorithm actually gives a 7-approximation for the minimum dominating set problem in unit-disk graphs, while adapting optimally to the mobility of the nodes in the network.

Recently, Kuhn et al. [Kuhn et al., 2004] presented a distributed algorithm that computes a constant factor approximation of a minimum dominating set in $O(\log^2 n)$ time without needing any synchronization but it requires that nodes know an estimate of the total number of nodes in the network. In [Parthasarathy and Gandhi, 2004], Parthasarathy and Gandhi also present distributed algorithms to compute a constant factor approximation to the minimum dominating set. The running time of their algorithm depends on the amount of information available to the nodes, and nodes have to know an estimate of the size of the network. Both papers extend the unit disk model taking into account signal interference.

2.2.2.1. SPANNERS

Suppose that we have a set of nodes V that are distributed in an arbitrary way in a Euclidean space. For $v, w \in V$, let $d(v, w)$ denote the Euclidean distance between v and w . The goal of the geometric spanner problem is to find a graph $G = (V, E)$ so that for each pair of

nodes $v, w \in V$ there is a path in G from v to w whose length is at most $t \cdot d(v, w)$ for some fixed constant t . In this case, G is called a *geometric t -spanner* of G where t is the stretch factor.

For geometric spanners, several structures have been proposed. Geometric spanners based on the Delaunay triangulation have been studied, e.g., [Gao et al., 2001, Li et al., 2002, Wang and Li, 2003]. Spanners based on the Yao graph [Yao, 1982] and Gabriel graph [Gabriel and Sokal, 1969] are presented in [Li et al., 2001, Wattenhofer et al., 2001, Song et al., 2004].

For topological spanners, Dubhashi et. al. [Dubhashi et al., 2003] presented a spanner with logarithmic stretch factor. Alzoubi et. al. [Alzoubi et al., 2003] presented a spanner with constant stretch factor of 5 where the protocol is very similar to ours but uses a high-level model for wireless networks.

Our protocol for selecting gateway nodes also has similarities to the protocols presented in [Wang and Li, 2002, Gao et al., 2001]. However both these papers are based on high-level wireless models.

2.2.3. Overview of Spanner Protocol

In the following, r_t denotes the desired transmission range and G_{r_t} represents the graph with node set V and edge set E_{r_t} containing all edges $\{v, w\}$ with $c(v, w) \leq r_t$.

Our spanner protocol for G_{r_t} consists of 3 phases:

- **Phase I:** The goal of this phase is to construct a constant density dominating set in G_{r_t} . This is achieved by extending Luby's algorithm [Luby, 1985] to our more com-

plex model. Since the dominating set resulting from phase I may not be connected, we need further phases to obtain a constant density spanner.

- **Phase II:** The goal of this phase is to organize the nodes of the dominating set of phase I into color classes that keep nodes with the same color sufficiently far apart from each other. Only a constant number of different colors is needed for this, where the constant depends on δ . Every node organizes its rounds into time frames consisting of as many rounds as there are colors, and a node in the dominating set only becomes active in phase III in the round corresponding to its color.
- **Phase III:** The goal of this phase is to interconnect every pair of nodes in the dominating set that is within a hop distance of at most 3 in G_{r_t} with the help of at most 2 gateway nodes, using the coloring determined in phase II to minimize interference problems. Constructions using gateway nodes were also presented in [Gao et al., 2001, Wang and Li, 2002] but assuming a higher level model of wireless networks.

Each phase has a constant number of time slots associated with it, where each time slot represents a communication step. Phase I consists of 3 time slots, phase II consists of 4 time slots, and phase III consists of 4 time slots. These 11 time slots together form a *round* of the spanner protocol (see also Figure 2). We assume that all the nodes are synchronized in rounds, that is, every node starts a new round at the same time step. As mentioned earlier, this may be achieved via GPS or beacons.

The spanner protocol establishes a constant density spanner by running sufficiently many rounds of the three phases. All of the phases are self-stabilizing. More precisely, once

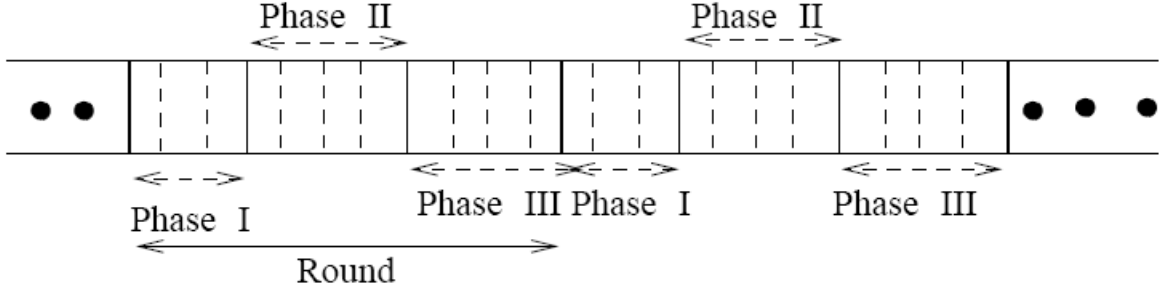


Figure 2. Two consecutive rounds of the spanner protocol.

phase I has selfstabilized, phase II will self-stabilize, and once phase II has selfstabilized, phase III will self-stabilize. In this way, the entire algorithm can self-stabilize from an arbitrary initial configuration.

It is not difficult to see that our spanner protocol results in a 5- spanner of constant density: Consider any pair of nodes s and t in G_{r_t} and let $p = (s = v_0, v_1, \dots, v_k = t)$ be the shortest path from s to t in G_{r_t} . Then we can emulate p via the connected dominating set by first going to a leader l_0 of s , then (possibly via gateway nodes) to a leader l_1 of v_1 , then to a leader l_2 of v_2 , and so on, until we reach a leader l_k of t , and finally to t . The length of this path is at most $3k + 2 \leq 5k$ for every $k \geq 1$. Combining this with the time bounds shown for the various phases in the sections below results Theorem 2.2.5.

Theorem 2.2.5 *For any desired transmission range r and any initial situation, the spanner protocol generates a constant density spanner in G_r in $O(D \log D \log n + \log^4 n)$ communication rounds, with high probability, where D is the maximum number of nodes that are within the transmission range of a node.*

An important feature of our protocol is that all messages sent are of constant length and the nodes only have to have a constant amount of storage, irrespective of the density

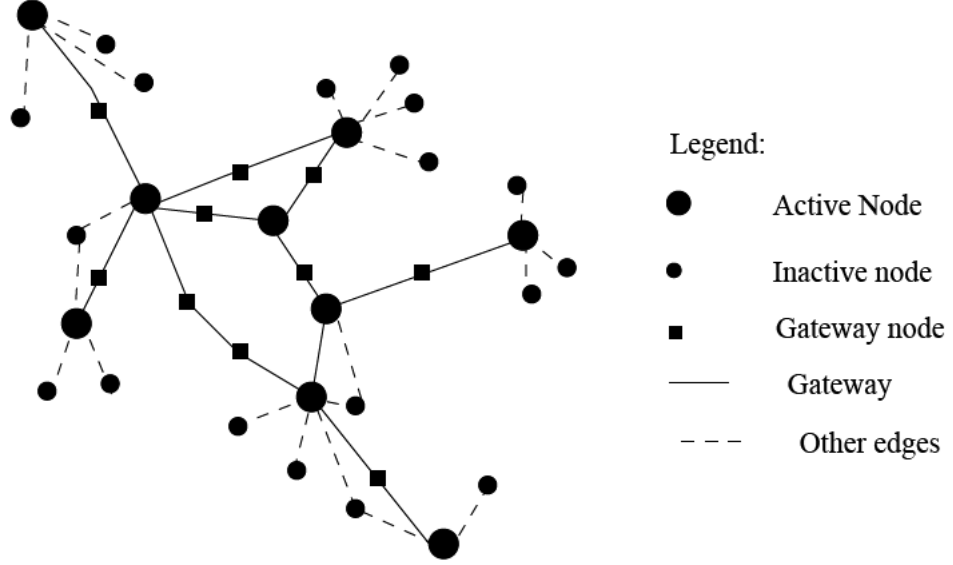


Figure 3. The spanner of the original network.

of the network. We just need the assumption that a storage unit is large enough to store the ID of a node. Hence, our protocol can be used with very simple devices such as sensors.

2.2.4. Phase I: Dominating Set

Let P be some fixed transmission power with transmission range r_t and interference range r_i for which we want to construct a dominating set of constant density. That is, given any set of nodes V , we want to find a subset $U \subset V$ of nodes so that every node $v \in V$ has at least one node $w \in U$ with $c(v, w) \leq r_t$ and at most some constant number of nodes $w \in U$ with $c(v, w) \leq r_t$.

As mentioned earlier, if we want to reach the goal above in a sub-linear number of steps without physical carrier sensing, then a good approximation of $\log n$ is needed, where $n = |V|$. Since our goal is to arrive at a dominating set without using any prior knowledge

of the network topology, physical carrier sensing has to be used, which complicates the design as it has uncertainties (see our model). To handle these uncertainties, we use a distributed coloring strategy together with two different sensing ranges.

In our protocol, nodes can either be *active* or *inactive*. The active nodes are the candidates for the dominating set. The nodes use two different sensing thresholds, depending on their state. The sensing threshold T_a has a CSI range of r_t and the sensing threshold T_i has a CST range of r_i . To distinguish between these ranges, we speak about an aCST/aCSI-range whenever we mean T_a and iCST/iCSI-range whenever we mean T_i .

Each node cuts the time into *time frames* of k *rounds* each for some constant number k that is the same for every node. The rounds are synchronized among the nodes but we do not require the frames to be synchronized.

Initially, all nodes are inactive. Afterwards, each node executes the Algorithm 1 in each round. In this protocol, each active node has exactly one, fixed active round in a frame and a signal is just a very simple message. Each item represents a communication step.

Algorithm 1 Phase I: Dominating Set

Each node v executes the following:

- 1: If v is active and in its active round, then v sends out an ACTIVE signal.
If v is inactive and v did not sense any ACTIVE signal for the last k rounds using a sensing threshold of T_a , v senses with threshold T_i , and if it does not sense anything, it becomes active and declares the current round number as its active round. If v did sense some ACTIVE signal in one of the last k rounds, it just performs sensing with threshold T_a and records the outcome.
 - 2: If v is active and is in its active round, then v sends out a LEADER message containing its ID with some fixed probability p (determined later). If v decides not to send out a LEADER message but it either senses a LEADER message with threshold T_a or receives a LEADER message, v becomes inactive.
-

In the following, let $H_{r,k} = (V, E)$ be an undirected graph that contains an edge

between two nodes v and w if and only if v and w are active and use the same active round (or color) k and $c(v, w) \leq r$. A node v is called a *leader* if it is active and there is no other active node w of the same color with $c(v, w) \leq r_t$. Since inactive nodes sense with an iCST range of r_i before they become active, none of the inactive nodes w with $c(v, w) \leq r_i$ will become active in the active round of v . Hence, we get:

Fact 2.2.6 *At any time, the set of leader nodes forms an independent set in $H_{r_t, k}$ that is disconnected from all other active nodes in $H_{r_t, k}$.*

In addition, a leader node uses an aCSI range of r_t and will therefore not be affected by nodes outside of a range of r_t . Hence, we arrive at the following fact.

Fact 2.2.7 *Once a node becomes a leader, it will stay a leader as long as the cost function c does not change.*

Furthermore, an inactive node v can only become active if in the previous k rounds there was no active node w with $c(v, w) \leq r_s$, where r_s is the CST range for threshold T_a , because otherwise v would have sensed the ACTIVE signal of w in one of these rounds. Hence, we also get:

Fact 2.2.8 *There cannot be two leaders v and w with $c(v, w) \leq r_s$.*

Since r_t/r_s is a constant, the facts above and Fact 2.2.1 imply that the leaders must form a set of constant density in G_{r_t} . On the other hand, the following lemma is true.

Lemma 2.2.9 *In any situation in which all active nodes are leaders but the leaders do not form a dominating set with respect to G_{r_t} , at least one inactive node will eventually become active.*

Proof. From Facts 2.2.1 and 2.2.8 it follows that there can be at most some constant number k' of leaders within the iCSI range of any node. Hence, if $k > k'$ then for every inactive node that does not yet have a leader within its transmission range there must be at least one round s in which there is no leader within its iCSI range. Because the inactive node will continue to explore potential active rounds in a round-robin fashion as long as it senses a transmission with threshold T_i , it will eventually arrive at round s and become active (unless some other inactive node close to it becomes active before that).

□

On the other hand, the following result is easy to check.

Lemma 2.2.10 *Every connected component of active nodes in $H_{r_t,k}$ results in at least one leader.*

Thus, the algorithm eventually arrives at a situation where there is no inactive node that does not have a leader within its transmission range. At that point, the leaders must form a superset of a maximal independent set in G_{r_t} . Thus, according to Facts 2.2.1, 2.2.8, and 2.2.7 the leaders eventually form a static dominating set of constant density. It remains to prove how much time is needed to reach such a state.

Theorem 2.2.11 *If all nodes are initially inactive, after $O(\log^4 n)$ rounds of the algorithm, the leaders form a static dominating set of constant density with respect to G_{r_t} , with high probability.*

Proof. The next two lemmata state important properties of connected components of active nodes in $H_{r_t,k}$. Notice that a leader always represents a connected component by itself.

Lemma 2.2.12 *At any time step t , $H_{r_t,k}$ consists of connected components of active nodes where all nodes in a connected component were reactivated at the same round.*

Proof. Suppose that there are two adjacent nodes, v and w , in some active, connected component in $H_{r_t,k}$ that were not reactivated at the same round. W.l.o.g. let v be the first node that became active. Then w could not have become active because v is in its iCST range, leading to a contradiction.

□

For the next lemma, given an active node v , we define $ls(v)$ as the bit sequence in which the i th bit is 1 if and only if v sent out a LEADER message in round i since it joined its current component. $ls(v)_i$ denotes the first i bits of $ls(v)$.

Lemma 2.2.13 *Every connected component of active nodes in $H_{r_t,k}$ needs at most $O(\log n)$ rounds, w.h.p., until every node in it either becomes inactive or becomes a leader.*

Proof. Consider any connected component C of active nodes in $H_{r_t,k}$ at some time point t_0 , and let C' be the union of the connected components of active nodes in $H_{r_t,k}$ that have at least one node within the interference range of a node in C .

Whenever a node becomes active after t_0 , it cannot interfere with the remaining nodes in C because it will be guaranteed to be outside of their interference range (and

therefore also of their aCSI range). Hence, we only need to focus on the remaining active nodes in $C \cup C'$.

We prove the lemma in two steps. First, we show that it only takes $O(\log n)$ rounds, w.h.p., until there are no two active nodes v and w in $C \cup C'$ where w is within the aCST range of v or vice versa. Then we show that it only takes $O(\log n)$ further rounds, w.h.p., until there are no two active nodes v and w in C that are within the transmission range of each other.

The probability that for any two fixed, active nodes v and w it holds that $ls(v)_i = ls(w)_i$ is equal to p^i . Hence, if $i = c \log_{1/p} n$, then the probability that there are two nodes v and w in $C \cup C'$ with $ls(v)_i = ls(w)_i$ that are within their aCST range is at most $n^2/p^{c \log_{1/p} n} = n^{2-c}$. Thus, the probability that after $c \log_{1/p} n$ rounds there are still two nodes within the aCST range in $C \cup C'$ that are both active is polynomially small in n for $c > 2$.

Hence, after $O(\log n)$ rounds, there can only be at most some constant number d of active nodes within the interference range of any active node in C , where d depends on the ratio between the interference range and the aCST range. Thus, when choosing $p = 1/d$, then the probability that exactly one of the active nodes within the interference range of an active node v in C is transmitting a LEADER message in a round is $\Theta(p)$. Therefore, it takes at most $O((1/p) \log_{1/p} n) = O(d \log_d n)$ rounds until for every node v in C that is still active there is no other active node in the transmission range of v , with high probability.

□

Next we give a lower bound on the number of leaders that emerge from a connected

component of active nodes in $H_{r_t, k}$. For the rest of the proof, we assume w.l.o.g. that $r_t = 1$ and $r_i = 1 + \alpha$ for some constant $\alpha > 0$. We define the area covered by an active node v as the area that is within the transmission range of v .

Lemma 2.2.14 *For any time step in which the currently existing connected components of active, non-leading nodes cover an area of $A = \Omega(\log^3 n)$, the number of leaders emerging from these components is $\Omega(A/\log^2 n)$, w.h.p.*

Proof. Consider any set C of connected components of active, non-leading nodes that cover an area of A . Given any node v , let $\Gamma(v)$ denote the set of nodes $w \in C$ with $c(v, w) \leq 1$ and let $\gamma(v) = |\Gamma(v)|$. Let H be the directed graph resulting from C by connecting two active nodes v and w by an edge (v, w) if and only if $c(v, w) \leq 1$ and $\gamma(w) \geq 2\gamma(v)$. A node is called a *sink* if it does not have any outgoing edges. H has the following important property:

Claim 2.2.15 *Every node v in H has a directed path to a sink s of length at most $\log n$.*

Proof. First of all, H cannot contain a directed cycle. Thus, every directed path must eventually end in a sink. Suppose now that some node v has a directed path p to a sink s of length more than $\log n$. Because of the definition of the edges, it follows that $\gamma(s) \geq 2^k \cdot \gamma(v) > n \cdot \gamma(v)$, which cannot happen because there are only n nodes in the system.

□

Recall that our cost function must satisfy $c(v, w) \in [(1 - \delta) d(v, w), (1 + \delta)d(v, w)]$. Thus, if we consider disks of radius $(1 + \log n)/(1 - \delta)$, around the sinks of H , then the

complete area A of active, non-leading nodes is covered. To extract out of all sinks a set of sinks useful for our analysis below, we consider these sinks one by one. For each sink s that has not already been eliminated, eliminate all sinks s' that are of distance at most 4 from s and add s to a set S . At the end, we arrive at a set S of sinks of pairwise distance at least 4 such that disks of radius $r = (5 + \log n)/(1 - \delta)$ around these sinks cover the entire area A . Thus, the area A can be decomposed into areas of size at most $a = \pi r^2$ each containing a sink in S , and therefore $|S| \geq |A|/a$. It is not difficult to show that these sinks have the following property:

Claim 2.2.16 *For any sink $s \in S$, the expected number of active nodes in $\Gamma(s)$ that become a leader is $\Theta(1)$.*

For any sink s , let the random variable X_s denote the number of active nodes in $\Gamma(s)$ that become leaders and let $X = \sum_s X_s$. From Claim 2.2.16 it follows that $E[X] \geq \alpha|S|$ for some constant $\alpha > 0$, and because the distance between any two sinks in S is at least 4, the X_s variables are independent. Thus, we can use Chernoff bounds to obtain

$$\Pr[X \leq (1 - \epsilon)\alpha|S|] \leq e^{-\epsilon^2\alpha|S|/2}$$

for any $\epsilon > 0$. This is polynomially small if $\epsilon = 1/2$ and $|S| = \Omega(\log n)$ is sufficiently large. Hence, in this case,

$$\Pr[X \leq \alpha|S|/2] = \Pr\left[X \leq \frac{\alpha|A|}{2\pi r^2}\right]$$

is polynomially small, which completes the proof of the lemma.

□

Now, let us call a node *unfinished* if it is active but not a leader or it is inactive and it does not have a leader within its transmission range. We know that an unfinished node is either active or must have at least one node within its iCSI range, r_{ii} , that was active within the previous k rounds (because otherwise it would become active). Hence, when drawing disks of radius $r_{ii}/(1 - \delta)$ around all nodes that were active in at least one of the k previous rounds, the entire area that the nodes can transmit messages to is covered.

Let A_0 be the area covered by the transmission ranges of all the nodes in the system. If $A_0 = \Omega(\log^3 n)$, then Lemma 2.2.13 and Lemma 2.2.14 imply that after $O(\log n)$ rounds, the area covered by the unfinished nodes is at most

$$A_0 - c \cdot \frac{A_0}{\log^2 n} = \left(1 - \frac{c}{\log^2 n}\right) A_0$$

for some constant c , with high probability. Thus, after k stages of $O(\log n)$ rounds each, the area covered by the unfinished nodes is at most

$$\left(1 - \frac{c}{\log^2 n}\right)^k A_0 \leq e^{(c \cdot k)/\log^2 n} A_0 ,$$

with high probability. The right hand side is less than $\log^3 n$ if $k \geq (\log A_0)(\log^2 n)/c$. Once an area of size $O(\log^3 n)$ is reached, it follows from Lemma 2.2.10 that it takes only $O(\log^3 n)$ more stages of $O(\log n)$ rounds each until there are no unfinished nodes any more. Since $A_0 = O(n)$, it follows that the total runtime needed for the set of active nodes to stabilize is $O(\log^4 n)$.

□

The dominating set algorithm can be easily extended so that it self-stabilizes

[Dijkstra, 1974] and it is robust against malicious behavior. Self-stabilization means that it can recover from *any* initial configuration.

2.2.4.1. SELF-STABILIZATION

An extra rule is necessary to provide self-stabilization because if the protocol above starts in a configuration violating Fact 2.2.8, it may not succeed in establishing a dominating set.

Consider adding a third step to each round of the protocol above. In this step, every active node sends a leader message with probability p and a transmission power so that its transmission range is only equal to the aCST range. Adding now the rule that whenever an active node receives a leader message in that step for a round different from its active round, then it becomes inactive, we do not have to assume anything about how the nodes are initially activated in order to satisfy Fact 2.2.8. So we get:

Corollary 2.2.17 *For any initial situation, the extended protocol needs at most $O(\log^4 n)$ rounds to arrive at a static dominating set of constant density with respect to G_{r_t} , w.h.p.*

2.2.4.2. ROBUSTNESS

Our dominating set algorithm is also highly robust against adversarial nodes. For any node v , let the $r_1 \oplus r_2$ -range of v be defined as the union of the r_2 -ranges of all the nodes within the r_1 -range of v . Given any distribution of nodes, let A be the area covered by the $r_{ii} \oplus r_t$ -ranges of adversarial nodes, where r_{ii} is the iCSI range of a node. Because in our protocol adversarial nodes can directly influence only nodes within their iCSI range, nodes beyond the r_t range of these nodes can only have leaders outside of A , and leaders outside of A will stay leaders forever, one can show:

Corollary 2.2.18 *If the honest nodes outside A are connected in G_{r_t} , then after $O(\log^4 n)$ rounds, the active honest nodes outside A form a dominating set of constant density with respect to G_{r_t} , w.h.p.*

2.2.5. Constant Density Spanner

In the next two subsections, we describe phases II and III in detail. We use the following notation. The constant d_1 refers to the number of active nodes that are within the interference range r_i of any node. The constant d_2 refers to the number of active nodes that are within the $r_i \oplus r_i$ -range of any node, and the constant g refers to the maximum number of required gateway connections for any active node. Finally, D refers to the density of the network, i.e. the maximum number of nodes within the transmission range of a node.

2.2.5.1. PHASE II - DISTRIBUTED LEADER COLORING

Similar to phase I, each node organizes the time into time frames consisting of cd_1 rounds for some constant c that is the same for every node. Also here, the rounds are synchronized but frames do not have to be synchronized among the nodes. We again assign active nodes to distinct rounds using a coloring mechanism. While the coloring in phase I was done with respect to G_{r_t} , we now need a coloring of the active nodes with respect to $G_{r_i \oplus r_i}$, that is, we need the active nodes to be at least $r_i \oplus r_i$ apart in order to receive the same color.

Every active node from phase I tries to own one of the rounds. An active node u is said to *own* a round if no other active node within its $r_i \oplus r_i$ range is using that round. Active nodes are in one of the states {owner, volatile}. An active node is in owner state if it already owns a round and is in volatile state if it is still trying to own a round. Active

nodes in owner state always send their ID in the first time slot of their round. Initially, every active node is volatile. Active nodes in volatile state choose an active round from the cd_1 possible rounds uniformly at random. Active nodes in owner state use a sensing threshold T_o with CST range r_i and active nodes in volatile state use a sensing threshold T_v with a CST range being equal to the CSI range of T_o , r_{ii} .

Active nodes do the following repeatedly. Every time a node re-activates, it sets its time stamp to 0. This time stamp is used by active nodes in Phase III to compare entries. Phase II algorithm is given at Algorithm 2.

Algorithm 2 Phase II - Distributed Leader Coloring

- 1: Every active node in owner state that is in its active round sends out a LEADER message containing its ID and its current time stamp and increases its time stamp by one afterwards.
- 2: Every active node in owner state that is in its active round decides with probability $1/2$ to send out an OWNER message either in the first or second substep of step 2.
- 3: Every inactive node that sensed a LEADER message with threshold T_v sends out a BUSY signal. Every active node in volatile state that senses a BUSY signal in its active round chooses a new active round uniformly at random.
- 4: Every inactive node that sensed OWNER messages in both substeps of step 2 with threshold T_o sends out a COLLISION signal.

If an active node in owner state senses a COLLISION signal and sent an OWNER message in the second substep, it changes into volatile state and chooses a new active round uniformly at random.

If an active node in volatile state did not sense a BUSY or COLLISION signal in its active round, it becomes an owner.

It is not difficult to show the following result:

Theorem 2.2.19 *Once a stable set of active nodes is available, it holds: If $c \geq 4$, then all active nodes will be in owner state after $O(\log n)$ rounds of the protocol, w.h.p.*

The theorem implies that after $O(\log n)$ rounds, all active nodes have chosen rounds

so that for any two active nodes ℓ and ℓ' with the same round and any inactive node v within the interference range of ℓ , ℓ' is outside of the interference range of v . Hence, ℓ can transmit messages to nodes within its transmission range without interference problems, and these nodes can transmit messages to ℓ without causing interference problems at ℓ . Both properties are important for phase III to work correctly.

Without the two types of signals BUSY and COLLISION and the two different sensing thresholds the coloring achieved may fail to be $r_i \oplus r_i$ distinct. For any active node ℓ in volatile state, the threshold T_v and the BUSY signal helps to identify the presence of active nodes in owner state with the same active round so that active nodes in owner state without another active node in owner state within the $r_{ii} \oplus r_{ii}$ -range will also keep this property in the future and are therefore safe from becoming volatile again. The COLLISION signal is necessary to resolve conflicts among close by active nodes in owner state with the same active round, which can happen if volatile nodes become an owner in the same round, or this may be part of the initial state when looking at self-stabilization. In any case, the monotonicity assumption on the sensing in our model is important to make sure that there will either never be a conflict among owner nodes or immediately a conflict when a collision is detected.

2.2.5.2. PHASE III - GATEWAY DISCOVERY

In this section we describe the protocol for phase III. The goal of this phase is for the active nodes from Phase I to discover gateway connections to other leaders that are within a hop distance of at most 3 in G_{r_t} .

During this phase, the active nodes use an aCST range of r_t . The active nodes

use the rounds reserved in phase II to achieve interference-free communication with the inactive nodes within their transmission range. Each round consists of four time slots for communication in phase III, where each time slot represents a communication step as shown in Figure 2. In the first time slot, inactive nodes send CLIENT messages and in the second time slot the active node sends a response accordingly; in the third and fourth time slots, an inactive node u may broadcast to its (active and inactive) neighbors all the information it has regarding possible gateways between the leader owning the reserved round and other leader nodes it has heard about. For simplicity, we assume that all active nodes are reactivated at the same time and hence that we can directly compare the time stamps with respect to the different active nodes. In reality, each inactive node u would keep track of the offsets of the (constant number of) time stamps it receives (in the corresponding slots allocated to the different leaders in phase II) and use these offsets when comparing time stamps from different leaders.

We first describe the data structures that are maintained during this phase. Each inactive node u maintains a cache, called $\mathcal{P}_u$, which has entries of the form (ℓ, v, t_ℓ) where ℓ is an active node, v is an inactive node (with $u = v$ possibly), and t_ℓ is the time stamp with respect to ℓ at which the entry (ℓ, v) is added to $\mathcal{P}_u$. When comparing entries in the cache, a $*$ acts as a wild card that matches any value. The operation $enqueue(\ell, v, t_\ell)$ on $\mathcal{P}_u$ is used to add the new entry (ℓ, v, t_ℓ) to $\mathcal{P}_u$. *Enqueue* performs the following checks before actually adding the new entry to $\mathcal{P}_u$. When adding a new entry (ℓ, v, t_ℓ) , any entry of the form $(\ell, *, t')$ with $t' < t_\ell$ is evicted. If no such entry exists and $\mathcal{P}_u$ is full, then the least recently added entry $(*, *, t')$, that is $t' = \min\{t | t < t_\ell \text{ and } (*, *, t) \in \mathcal{P}_u\}$, is evicted

to make room for the new entry. The cache $\mathcal{P}_u$ has space enough to store a constant, d_2 , number of entries. Inactive nodes also maintain a state that is either *awake* or *asleep* with respect to each active node that is within their transmission range. The *asleep* nodes just listen the channel and becomes *awake* when they receive a FREE or a ACK message.

Each active node ℓ maintains a list, called $\mathcal{G}_\ell$, and each entry in $\mathcal{G}_\ell$ contains two fields. The first field has gateways represented as quadruples of the form (ℓ, u, v, ℓ') where $\ell' \neq \ell$ and $u = v$ possibly, ℓ' is an active node and u, v are inactive nodes. The second field contains the time stamp t_ℓ at which the entry was added to $\mathcal{G}_\ell$. The operation *enqueue* on $\mathcal{G}_\ell$ is used to add a new entry $((\ell, u, v, \ell'), t_\ell)$ to $\mathcal{G}_\ell$. Before adding the new entry $((\ell, u, v, \ell'), t)$ to $\mathcal{G}_\ell$, any entry of the form $((\ell, *, *, \ell'), t')$ is evicted from $\mathcal{G}_\ell$ for $t' < t$. If the list $\mathcal{G}_\ell$ is full, then the entry corresponding to t' such that $t' = \min\{t'' | t'' < t \text{ and } ((\ell, *, *, \ell'), t'') \in \mathcal{G}_\ell\}$, that is the entry of $\mathcal{G}_\ell$ with smallest time stamp, is deleted to make room for the new entry. (Similar to *enqueue* on $\mathcal{P}_u$ for inactive node u). The list $\mathcal{G}_\ell$ has space enough to store a constant, g , number of entries.

In Algorithm 3, ℓ refers to the ID of the active node that owns the current slot and u is an inactive node that received the ID message from ℓ and the state of u is with respect to ℓ .

Before we analyze the protocol, we start with the following fact, which follows from the observation that a necessary condition for an inactive node u to transmit in step 3 and step 4 is to receive an ACK from an active node in step 2.

Fact 2.2.20 *During steps 3 and 4 of the protocol there are at most a constant number d_1 of nodes that are transmitting any message.*

Algorithm 3 Phase III - Gateway Discovery

- 1: If u is awake then u sends out a CLIENT message of the form $\langle \text{CLIENT}, \ell, u \rangle$ with probability $1/2$.
 - 2: Node ℓ responds with a reply in the next time slot which can be of three forms. If ℓ receives a CLIENT message from node u then ℓ adds u to N_ℓ by calling $\text{enqueue}(u)$ and also sends an acknowledgment containing the ID of u as $\langle \ell, \text{ACK}, u \rangle$. If ℓ only senses a busy channel but does not receive any message, then ℓ sends a collision message of the form $\langle \ell, \text{COLLISION} \rangle$. If ℓ does not receive any message and also does not sense a busy channel, the ℓ sends a free channel message of the form $\langle \ell, \text{FREE} \rangle$.
If u is awake and decided not to send a CLIENT message in the previous slot and receives a collision message then u goes to asleep state. If u is asleep and receives a free channel or an acknowledgement message then u becomes awake.
 - 3: If u is awake and receives an acknowledgment containing the ID of u then u will store (ℓ, u, t_ℓ) in $\mathcal{P}_u$, where t_ℓ is the current time stamp associated with ℓ , by calling $\text{enqueue}(\ell, u, t_\ell)$.
Node u also deletes any entries of the form $(*, \ell)$ from $\mathcal{P}_u$ (since ℓ is no longer inactive). Node u then broadcasts, in the third time slot, a message $\langle \text{ADV}, \ell, u, t_\ell \rangle$ to its neighbors.
 - 4: Node u builds one GATEWAY message containing all quintuples of the form $((\ell, u, v_j, \ell_j), t)$ for each j such that $\ell_j \neq \ell$ with $(\ell_j, v_j, t_j) \in \mathcal{P}_u$, where $t = \min\{t_\ell, t_j\}$, and sends the message to its neighbors. The GATEWAY message is sent in the fourth time slot.
If v is not active and received an ADV message $\langle \text{ADV}, \ell, u, t_\ell \rangle$ then it calls $\text{enqueue}(\ell, u, t_\ell)$ on $\mathcal{P}_v$. Node v also deletes any entries of the form $(u, *)$ or $(*, \ell)$ from $\mathcal{P}_v$ (as u is no longer an active node nor is ℓ inactive).
If ℓ is active and receives a GATEWAY message containing $((\ell, u, v, \ell'), t)$, then ℓ stores $((\ell, u, v, \ell'), t)$ in $\mathcal{G}_\ell$ by calling $\text{enqueue}((\ell, u, v, \ell'), t)$.
-

Using this fact, we can prove the following theorem.

Theorem 2.2.21 *In $O(D \log n \log D)$ rounds, each active node learns about a gateway to each of the currently active nodes in its 3-neighborhood with respect to G_{r_t} , w.h.p.*

Proof. We prove the convergence of phase III to a set of valid gateway connections in $O(D \log n \log D)$ rounds after phase I and phase II have reached a stable state. Since, at that point the active nodes have reserved rounds that are distinct within the $r_i \oplus r_i$ range, we can treat the actions of active nodes independent of each other.

Let (v, ℓ) be an inactive node-active node pair such that v has to send a CLIENT message to ℓ . Node v has at most $O(D)$ inactive nodes in its interference range sending a CLIENT message to some leader node. If more than one node in awake state, with respect to ℓ , decides to send a CLIENT message, then ℓ will send a collision message. Since the collision message will be received by the inactive nodes, within r_t range of ℓ , awake nodes that decided not to send a CLIENT message to ℓ in the previous slot will go to asleep state.

Consider time to be partitioned into groups of consecutive rounds such that each group ends with a round where the active node ℓ sends either an ACK message or a FREE message. (A group ending with an ACK message signifies a successful group and a group ending with a FREE message is a failed group). Notice that at the end of every group, whether successful or not, all the inactive nodes within the r_t range of ℓ go to awake state (by step 2 of the protocol).

It is not difficult to show that the expected number of rounds in each group, successful or failed, is $O(\log D)$ and any group is successful with constant probability. Due to symmetry reasons any inactive node is equally likely to be send a CLIENT message in a successful group. Thus, during any successful group, for a given pair (v, ℓ) ,

$$\Pr[v \text{ sends a CLIENT message successfully to } \ell] \geq 1/cD$$

for some constant $c > 1$.

Using Chernoff bounds, for any given pair (v, ℓ) the probability that it takes more than Dk groups so that v sends a CLIENT message to ℓ successfully will be polynomially small for $k = O(\log n)$. It can also be shown that each group has $O(\log D)$ rounds not only on expectation but also with high probability. Thus any node v requires at most

$O(D \log n \log D)$ rounds to send a CLIENT message to ℓ successfully w.h.p.

To proceed further, let ℓ and ℓ' be active nodes, with $d(\ell, \ell') \leq 3$ and let (ℓ, u, v, ℓ') be a gateway between ℓ and ℓ' . Notice that once ℓ and ℓ' receive CLIENT messages from u and v respectively, ℓ and ℓ' can establish a gateway connection between them as successful CLIENT messages are followed by ADV and GATEWAY messages in the next time slots reserved for this phase. Without loss of generality, we assume that u sends the ADV message that v receives and adds the entry (ℓ, u, v, ℓ') to the GATEWAY message that v sends. Along with Fact 2.2.20 it holds that during every group the probability that u gets an ACK message and sends the ADV message is $\geq 1/c'D$ for a constant $c' > 1$. And similarly the probability that v gets an ACK message from ℓ' and sends a GATEWAY message is $\geq 1/c'D$. Thus, in each group,

$$\Pr[\ell \text{ and } \ell' \text{ discover a gateway connection}] \geq 1/c''D$$

for some constant $c'' > 1$. Using calculations similar to the above, it holds that ℓ and ℓ' can establish a gateway connection in $O(D \log n \log D)$ rounds w.h.p.

Note that, after phase II stabilizes and after we let phase III run for $O(D \log n \log D)$ time steps, time stamping will be enough to guarantee that we will always keep information received at a leader node ℓ about a valid gateway connection between leader nodes ℓ and ℓ' , if at least one such connection exists (since we have at most a constant number of leader nodes within cost $3r_t$ from any given leader node, and since we have at most a constant number of leader nodes adjacent to any inactive node, constant size $\mathcal{P}_u$ and $\mathcal{G}_\ell$ lists at inactive nodes u and active nodes ℓ respectively will suffice).

□

2.3. Broadcasting and Information Gathering in Wireless Ad-Hoc Networks

In this section, we will design broadcasting and information gathering algorithms for wireless ad hoc networks[Onus et al., 2005, Kothapalli, 2006]. These algorithms use the communication model presented in the previous section. And these algorithms are build on top of the overlay network designed in the previous chapter.

The problem of broadcasting can be described as follows. Given a static connected wireless network of n nodes, minimize the total time and work to send $m \geq 1$ broadcast messages originating from a source node s to all nodes in the network. In Section 2.3.1, we consider the simple case where a single node s is the source of a single broadcast message, i.e., $m = 1$. In Section 2.3.2, we extend our algorithm to handle the case that node s is the source of multiple broadcast messages.

Information gathering is another important communication primitive in wireless networks. The problem has applications in many scenarios in sensor networks [Intanagonwiwat et al., 2000, Yu et al., 2004, Enachescu et al., 2004], and maintaining connectivity with base stations in a multihop wireless network. The problem of information gathering can be described as follows. Given a static connected wireless network of n nodes among which m packets are arbitrarily distributed and a sink node s in the network, minimize the total time and work required for sending the m packets to the sink node. In Section 5, we present and analyze a simple strategy for information gathering.

Our algorithms are self-stabilizing (i.e., can start in an arbitrary state) and can therefore adapt to changes in a wireless ad-hoc network. Our algorithms do not require any knowledge of the size of the network. For our algorithms to work correctly, it suffices that

the nodes in the network have identifiers that are locally different. We only require that the nodes synchronize locally into rounds up to some reasonably small time difference, which can be easily accomplished using GPS signals or any form of beacons. Another important feature of our algorithms is that a constant amount of storage at any node suffices even in the case of gathering. The above properties make our algorithms applicable to sensor networks without any modifications.

2.3.1. Isolated Broadcasting

In this section we consider the problem of broadcasting a single message. Let node s be the source of the broadcast message. Since s has a maximum distance of $D(s)$ to any node in G_{r_t} , $D(s)$ is a lower bound on the time an optimal offline algorithm needs to broadcast a message from s to all nodes. Our goal is to come up with a broadcast scheme so that the time needed by the broadcast message to reach all nodes is as close to $D(s)$ as possible. We use the constant density spanner construction of [Kothapalli et al., 2005] as the basis. If s is not an active node, i.e., $s \notin U$, then let l be some active node that is within the transmission range of s . Then s first sends the message to l . The broadcast scheme then proceeds in rounds that are synchronized among the nodes. In the broadcast scheme below, l refers to the ID of an active node that owns the current slot. Every item in the algorithms is a separate time step.

Notice that inactive nodes just need to listen to the wireless channel in order to receive the broadcast message eventually. This is because our spanner algorithm in this chapter makes sure that message transmissions of active nodes in step 1 above never inter-

Algorithm 4 Isolated Broadcasting

- 1: If l received the broadcast message in the previous round and it is the first time it received the broadcast message, l sends out the broadcast message.
 - 2: If v is a gateway node and has already received the broadcast message, then v sends out an RTS (Request- To-Send) signal with probability p .
 - 3: If v is a gateway node and decided not to send out an RTS signal or v is an active node, then v checks if it correctly received an RTS signal. If so, and v has not received the broadcast message yet, v sends out a CTS (Clear-To-Send) signal.
 - 4: If v is a gateway node and sent out an RTS signal, then v checks if it sensed a CTS signal. If so, v sends out the broadcast message.
-

fere at an inactive node.

2.3.1.1. TIME EFFICIENCY

The following theorem demonstrate that the above protocol has a high time efficiency.

Theorem 2.3.1 *Given the constant density spanner of G_{r_t} as in this chapter, the broadcast algorithm with $p = 1/\hat{d}$ needs $O(D(s) + \log n)$ rounds, with high probability, to deliver the broadcast message to all nodes.*

Proof. Recall that every message has to traverse a path of length at most $5D(s)$ to reach any node via the nodes in $U \cup G''$. Hence, consider any fixed node v , and let P be any shortest path from s to v via the dominating set. Let $\alpha \leq 5D(s)$ be the length of P . Suppose that it takes at least $\alpha + \delta$ time steps to send the broadcast message to v . Then there must have been at least δ time steps in which a node in P failed to forward the message to the next node in P , and the time steps can be associated with nodes of monotonically increasing order in P (by always looking at the node of maximum distance from s in P that failed to forward the packet at any given time step). The number of possibilities to assign δ out of the $\alpha + \delta$ time steps to nodes in P in increasing order is $\binom{\alpha + \delta}{\delta}$, which is equal to the number

of all binary sequences of length $\alpha + \delta$ with δ 1's. A 0 represents the event “move to the next node in P ” and a 1 represents the event “failed transmission” at the current node in P .

Since the probability of a node v to fail is equal to

$$\begin{aligned} & \Pr[v \text{ did not decide to transmit the message}] + \Pr[v \text{ transmitted the message but failed}] \\ & \leq (1 - p) + p(1 - (1 - p)^{d_1 - 1}) = (1 - p) + p(1 - (1 - 1/d_1)^{d_1 - 1}) \leq 1 - p + p(1 - 1/e) \\ & = 1 - p/e, \end{aligned}$$

it follows that the probability that the broadcast message needs more than $\alpha + \delta$ steps with $\delta = b \cdot \alpha$ is at most

$$\binom{\alpha + \delta}{\delta} (1 - p/e)^\delta \leq \left(\frac{e(\alpha + \delta)}{\alpha} \right)^\alpha \cdot (1 - p/e)^\delta \leq (e(1 + b)e^{-pb/e})^\alpha$$

which is polynomially small in n if $b = \Omega((1 + \log n/\alpha)/p)$. Thus, it takes $O(D(s) + \log n)$ time steps until the broadcast message reaches any particular node v , w.h.p., which completes the proof of the theorem.

□

2.3.1.2. WORK EFFICIENCY

Next we consider the work efficiency of the broadcast algorithm. We assume that the cost for sending and sensing the RTS/CTS message is negligible so that we only need to bound the cost for retransmitting the broadcast message.

Suppose that the area covered by disks of radius r_t around all nodes in the system is A . Then a lower bound for the number of message transmissions of an optimal offline

algorithm is $|A|/(4\pi r_t^2)$. Next we prove an asymptotically matching upper bound for our broadcast algorithm. We show the following theorem.

Theorem 2.3.2 *Given the constant density spanner of G_{r_t} as described in this chapter, the broadcast algorithm needs $O(|A|/r_t^2)$ work to send a broadcast message from s to all nodes.*

Proof. Due to the constant density of the network of active nodes, we need $\Omega(|A|/r_t^2)$ transmissions of the broadcast message in order to reach all active nodes. Each active node has a fixed time slot, and thus one transmission from each active node is enough for all inactive nodes to receive the message. Since the spanner construction in this chapter results in a spanner of constant density, $O(|A|/r_t^2)$ messages will suffice to deliver the broadcast message to all inactive nodes.

It remains to show that the work necessary for the active nodes to receive the message (from gateway nodes) is also $O(|A|/r_t^2)$. Note that an active or gateway node only sends a CTS message if a broadcast message can be sent to it without interfering with other transmissions. Since every active or gateway node sends a CTS at most once, the number of message transmissions necessary for all active or gateway nodes to receive the message is equal to the total number of active and gateway nodes, which is $O(|A|/r_t^2)$.

Thus, the broadcast algorithm will send $O(|A|/r_t^2)$ messages to send the broadcast message to all the nodes in the network.

□

2.3.1.3. SELF-STABILIZATION

The broadcast algorithm can also provide self-stabilization. We add a fifth step to the broadcast algorithm in which if a node v just woke up or moved, then v sends an AWAKE signal. Furthermore, we replace the first step with the following step:

- If ℓ is active and received the broadcast message in the previous round and it is the first time it received the broadcast message or if ℓ sensed an AWAKE signal in the previous round or if ℓ just became active in this round, ℓ sends out the broadcast message.

2.3.2. Broadcasting Multiple Messages

Next we look at the case that the source s wants to send out multiple broadcast messages instead of just one. Then s attaches continuous sequence numbers to the messages, starting with 1.

The broadcast scheme proceeds in rounds that are synchronized among the nodes. Each active or gateway node v keeps track of two numbers, i_v and j_v . Number i_v denotes the minimum message number v has not received so far and number j_v denotes the minimum message number (v knows about since its last successful transmission attempt) a node of distance at most r_t from v has not received so far. In the broadcast scheme below, l refers to the ID of an active node that owns the current slot. Initially, for each gateway and active node v , $i_v = j_v = 1$. In each round, every node $v \neq s$ does the following. Each item below represents a separate time step.

The source node s uses the same protocol as above with the only difference that it

Algorithm 5 Broadcasting Multiple Messages

- 1: If l received the broadcast message with sequence number $i' = i_l$ in the previous round, then it sets $i_l = i_l + 1$ and sends out the broadcast message with sequence number i' .
If v is a gateway node and received a broadcast message with sequence number $i' = i_v$, then it sets $i_v = i_v + 1$.
 - 2: If v is an active or gateway node, then it sends out an (RTR, i_v) message (RTR means ready-to-receive) with probability p . If v decides not to send out an RTR message, it checks whether it is able to receive an (RTR, i') message. If so, it sets $j_v = \min\{j_v, i'\}$.
 - 3: If v is a gateway node and $i_v > j_v$, then it sends out an (RTS, j_v) message with probability p .
If v is a gateway node and decided not to send out an RTS message or v is an active node, then v checks if it correctly received an (RTS, j') message with $j' = i_v$. If so, v sends out a CTS signal.
 - 4: If v is a gateway node and sent out an (RTS, j_v) message, then v checks if it sensed a CTS signal. If so, v sends out the broadcast message with sequence number j_v . Afterwards, v sets $j_v = \min\{j_v + 1, i_v - 1\}$. If v is a gateway node and did not send a message but received a broadcast message with sequence number $i' = i_v$, then it sets $i_v = i_v + 1$.
-

only executes the first step. The inactive nodes just need to listen to the wireless channel in order to receive the broadcast messages eventually. The following theorems demonstrate that this protocol has a high time and work efficiency.

2.3.2.1. TIME EFFICIENCY

The following theorem demonstrates that this protocol has a high time efficiency. Recall that d_1 refers to the number of active nodes that are within the interference range, r_i , of any node.

Theorem 2.3.3 *Given the constant density spanner of G_{r_i} as described this chapter, the multiple broadcast algorithm with $p = 1/d_1$ needs $O(D(s) + m + \log n)$ rounds, with high probability, to deliver m broadcast messages to all nodes.*

Proof. The proof involves substantial extensions to the proof of Theorem 2.3.1.

We use a delay sequence argument to prove the theorem. For simplicity, we assume each round of the algorithm takes one time unit. Let v be the last node that received all broadcast messages in $U \cup G''$, let $P = (u_0 = s, u_1, \dots, u_d = v)$ be any shortest path of active and gateway nodes from s to v , and let $\alpha \leq 5D(s)$ be the length of P . Suppose that it takes t_m time steps for all broadcast messages to reach v . Given a node w , let $m(w)$ denote the number of messages node w has already received. We go backwards in time from the moment the last message reached v and move monotonically backwards along P from $v_m = v$ so that at any time we stay at a node u_j with

- $m(u_j) = m$,
- $m(w) = m - 1$ for some node w within range r_t of u_j , and
- $m(w) \geq m - 1$ for all nodes w within range r_t of u_j .

Such a node is called m -active. This is done until we reach the first time step t_{m-1} at which there is a node $u_{j'}$ with $j' \leq j$ that is not m -active. We then set $u_{j'} = v_{m-1}$. An m -active node must always exist at any time from $t_{m-1} + 1$ to t_m because within these steps, every node from s to the current u_j must have only neighbors w with $m(w) \geq m - 1$, s has all m messages, and at step t_m , u_{d-1} has initially at least one neighbor w with $m(w) = m - 1$, namely v .

From step t_{m-1} , we go backwards in time and move monotonically backwards along P from v_{m-1} so that at any time we stay at a node u_j that is currently $m - 1$ -active. This is done until we reach the first time step t_{m-2} at which there is a node $u_{j'}$ with $j' \leq j$ that is

only $m - 2$ -active. We then set this $u_{j'} = v_{m-2}$. Again, an $m - 1$ -active node must always exist at any time from $t_{m-2} + 1$ to t_{m-1} because within these steps, every node from s to the current u_j must have only neighbors w with $m(w) \geq m - 2$, s has all m messages, and at step t_{m-1} , the node before v_{m-1} has initially at least one neighbor w with $m(w) = m - 2$, namely v_{m-1} .

Continuing with this argument gives us a sequence of nodes $v_1, \dots, v_m$ of monotonic order from s to v and a sequence of time steps $1 \leq t_1 < t_2 < \dots < t_{m-1} < t_m$ with the property that between $t_i + 1$ and t_{i+1} there is a monotonic sequence of $i + 1$ -active nodes from v_i to v_{i+1} . For any i , at most $d_1 \cdot l_i$ of the time steps from $t_i + 1$ to t_{i+1} can represent successful transmissions (i.e. message $i + 1$ was successfully sent from u_j to some active or gateway node w within range r_t of u_j), where l_i is the distance along P from v_i to v_{i+1} . This is because every active or gateway node between v_i and v_{i+1} has at most d_1 other active or gateway nodes.

Hence, altogether, there can be at most $d_1 \alpha$ successful transmissions. All others must have failed. All transmissions of active nodes are successful, since they have fixed time slots. For gateway nodes, the probability of a transmission to fail is at most

$$\begin{aligned}
& \Pr[u_j \text{ did not decide to send a message}] + \\
& \Pr[u_j \text{ decided to send a message with an incorrect sequence number}] + \\
& \Pr[u_j \text{ decided to send a message but failed}] \\
& \leq (1 - p) + p((1 - p) + p(1 - (1 - p)^{d_1 - 1})) + (p(1 - (1 - p)^{d_1 - 1})) \\
& \leq 1 - (p/9)
\end{aligned}$$

Thus, the probability that the broadcast message needs more than $S + \delta$ steps, $S = d_1\alpha + m$ with $\delta = b \cdot S$ is at most

$$\binom{S + \delta}{\delta} \binom{S}{\alpha} (1 - p/9)^\delta \leq \left(\frac{e(S + \delta)}{S} \right)^S \left(\frac{eS}{S - \alpha} \right)^{S - \alpha} (1 - p/9)^{b \cdot S} \leq (3e(1 + b)e^{-pb/9})^S$$

This is polynomially small in n if $b = \Omega((1 + \log n/S)/p)$. Thus, in $O(D(s) + m + \log n)$ time steps all broadcast messages reach all active and gateway nodes, with high probability. Since each active node has a fixed time slot, m transmissions from each active node is enough for inactive nodes to receive all messages. Thus, in $O(D(s) + m + \log n)$ time steps, all nodes will receive all m messages, w.h.p..

□

2.3.2.2. WORK EFFICIENCY

The following theorem demonstrates that this protocol has a high time efficiency.

Theorem 2.3.4 *Given a constant density spanner of G_{r_t} as described in this chapter, the multiple broadcast algorithm needs $O(m|A|/r_t^2)$ work to send all m broadcast messages to all nodes in the system.*

Proof. The proof is very similar to the proof of Theorem 2.3.2. A lower bound on the number of message transmissions of an optimal offline algorithm is $\Omega(m|A|/r_t^2)$. Similarly, $O(m|A|/r_t^2)$ messages will be sufficient to deliver m broadcast messages to all inactive nodes and $O(m|A|/r_t^2)$ messages will be sufficient to deliver m broadcast messages to all active nodes. Thus, the broadcast algorithm will need $O(m|A|/r_t^2)$ messages w.h.p., to send m broadcast messages to all nodes in the system.

□

2.3.2.3. SELF-STABILIZATION

The multiple broadcast algorithm can also provide self-stabilization. We can add a fifth step to the broadcast algorithm in which if a node v just woke up or moved, then v sends an AWAKE signal. Each active node v also keeps track of number k_v , which is initially equal to 1. Number k_v denotes the minimum message number a newly awake or arrived node of distance at most r_t from v has not received so far. We replace the first step with the following step:

- If ℓ received the broadcast message with sequence number $i' = i_\ell$ in the previous round, then it sets $i_\ell = i_\ell + 1$ and sends out the broadcast message with sequence number i' .

If ℓ sensed an AWAKE signal in the previous round, it sets $k_\ell = 1$. If $k_\ell < i_\ell$, ℓ sends out the broadcast message with sequence number k_ℓ and sets $k_\ell = k_\ell + 1$.

2.3.3. Information Gathering

We now consider the situation where a total of m packets distributed in an arbitrary way among the nodes in the wireless network are to be delivered to a sink node s in the network. Firstly, note that $\Omega(m + D(s))$ is a lower bound on any solution for the information gathering problem. In the following, we describe a 2-stage protocol to perform information gathering efficiently. Each stage has a constant number of reserved time slots, 4 slots for stage 1 and 4 slots for stage 2.

2.3.3.1. STAGE 1: BUILDING GATHERING TREE $T(s)$

We first show how to build the gathering tree rooted at s . All internal nodes in this tree will belong to $U \cup G$. The sink node s , if it is not in $U \cup G$, selects an active node l such that $d(l, s) = 1$ and sends a route packet to l with sequence number of 0 of the form $\langle ROUTE, s, 0 \rangle$. The rest of the nodes do the following. Initially, $d'(s, v) = \infty$, $\pi(v) = NULL \forall v \in U \cup G$. Let $d'(s, s) = 0$ and $\pi(s) = s$.

Algorithm 6 Stage 1: Building Gathering Tree $T(s)$

- 1: If $u \in U \cup G$ receives a message $\langle ROUTE, v, d'(s, v) \rangle$ from v with a sequence number of $d'(s, v)$ and if $d'(s, v) + 1 < d'(s, u)$, then u sets $\pi(u) = v$ and $d'(s, u) = d'(s, v) + 1$. Node u also sets $\text{flag}(u) = 1$ in this case, indicating that u has to send a route message since u updated its predecessor. If $u \notin U \cup G$ and $d'(s, v) + 1 < d'(s, u)$ and $v \in U$, then u updates $\pi(u) = v$ and $d'(s, u) = d'(s, v) + 1$.
 - 2: If $u \in U \cup G$ and $d'(s, u) \neq \text{inf}$ and $\text{flag}(u) = 1$ then u sends an RTS signal with probability p to be determined later.
 - 3: If $u \in U \cup G$ and u received an RTS signal then u sends a CTS signal.
 - 4: If $v \in U \cup G$ and v sent an RTS signal and receives a CTS signal then v sends a route packet $\langle ROUTE, v, d'(s, v) \rangle$ and sets $\text{flag}(v) = 0$ signifying that the update has been notified.
-

The above construction has the following properties. Let $T' = (V', E')$ be a graph with $V' = U \cup \{s\} \cup \mathcal{G}$ and $E' = \{(v, \pi(v)) | v \in V'\}$. Then, T' is a shortest path tree rooted at s for the graph $G'' = (V'', E'')$ with $V'' = V'$ and $E'' = E \cap ((U \cup \{s\}) \times \mathcal{G})$. We set $T(s) = (V_T, E_T)$ with $V_T = V$ and $E_T = \{(v, \pi(v)) | v \in V_T\}$ where $\pi(v)$ is set as in step (1) of the above protocol. Since the graph G' defined in this chapter is a 5-spanner of the original network G_{r_t} , it also holds that $\max_{v \in V} d_{T(s)}(s, v) \leq 5 \max_{v \in V} d(s, v)$. Thus the maximum distance of any node from s in $T(s)$ is only a constant factor away from the maximum distance of any node from s in G .

Finally, we show the following lemma for the time steps required to construct $T(s)$

using the above protocol.

Lemma 2.3.5 *Given a connected dominating set of active and gateway nodes with constant density d_1 in G_{r_t} , the protocol to construct the gathering tree $T(s)$ given above takes at most $O(\log n + D(s))$ time steps w.h.p. when the probability p is set to $1/d_1$.*

Proof. The proof follows from the following claim.

Claim 2.3.6 *After $O(d + \log n)$ time steps it holds that w.h.p., all nodes $v \in U \cup \mathcal{G}$ that are at a distance d from s in G'' have their predecessor pointer $\pi(v)$ such that the path from v to s has length of d .*

Proof. The proof of the claim follows by induction on d starting from $d = 0$. Here only s is the node that has a distance 0 from s and the claim holds in this case due to the initialization. For the induction step, assume that all nodes with distance d have their predecessor correctly assigned. For any node u at distance $d + 1$ from s in G'' let it take more than $(d + 1) + \delta$ time steps to have $\pi(u)$ set correctly. Then in a path P of length $d + 1$ from u to s , we can associate δ nodes that are monotonic in distance from s that failed to forward a route packet. Using calculations similar to that of Theorem 2.3.1, then a value of $\delta = O(\log n)$ suffices to get the claim hold w.h.p.

□

Once we have the above claim, then the lemma follows as the maximum distance of any active or gateway node from s is $D(s)$. Inactive nodes do not have to send any route messages but can use step 1 of the protocol to set their predecessor pointer.

□

2.3.3.2. STAGE 2: GATHERING ON $T(s)$

In the gathering tree, $T(s)$, constructed in stage 1, each node has an unique path to the sink node s via the predecessor pointers π . Nodes use this path system to eventually deliver packets to s .

The active node uses the first time slot to deliver packets and the second and third time slots are used to coordinate the actions of the inactive nodes. Nodes $l \in U \cup G$ have a queue, Q_l , which can hold a constant number of packets. This queue works as a first-in-first-out list and supports operations *enqueue* and *dequeue* which add a packet and return a packet respectively to Q_l .

In the following when we refer to inactive nodes, it is implicit that we are referring to those inactive nodes that have a packet to send. Inactive nodes have a state among *awake*, *asleep*. Initially all inactive nodes are in the *asleep* state.

We show in the following that the gathering protocol described above is efficient in terms of the time and work, and also it suffices for nodes in $U \cup G$ to have a queue that can store a constant number of packets in transit. We use the parameter Δ_m defined to be the density of nodes that have a packet to send. That is, Δ_m refers to the maximum number of nodes that are within a disk of radius r_t around any node that has a packet to send to s .

Lemma 2.3.7 *For any inactive node v , it takes $O(\Delta_m \log^2 n)$ time steps w.h.p, until v sends the packet to an active node that is within the transmission range of v .*

Proof. The proof is similar to that of the proof of Theorem 2.2.21. Given a stable gathering tree, the active nodes have reserved rounds that are distinct within the $r_i \oplus r_i$ range, we

Algorithm 7 Stage 2: Gathering on $T(s)$

- 1: If l is active and has a non-empty queue, then l sends the packet $dequeue(Q_l)$ during the time slot owned by l . This packet has a destination $\pi(l)$ and nodes other than $\pi(l)$ discard the packet and $\pi(l)$ stores the packet by calling *enqueue* on $Q_{\pi(l)}$. In the second time slot, the active nodes listen to the channel.
 If g is a gateway node and has a non-empty queue then g sends an RTS message containing the id of $\pi(g)$ with probability p , where p is to be determined later.
 - 2: If $u \in U \cup G$ and Q_u is not full and u receives an RTS message containing the id of u then u sends a CTS message.
 If u is inactive and has a packet to send and u is *awake* then u sends an I-RTS (for *Inactive-RTS*) signal to $\pi(u)$ with a probability $1/2$.
 - 3: If g is a gateway node and sent an RTS signal in the previous time step and receives a CTS signal from $\pi(g)$ then g sends the packet $dequeue(Q_g)$ to $\pi(g)$. This packet has a destination $\pi(g)$ and other nodes that receive the packet ignore it.
 If l is active and l receives an I-RTS signal from an inactive node u then l sends an I-CTS signal. If l senses a busy channel but does not receive any I-RTS signal, then l sends a collision message of the form $\langle l, COLLIDE \rangle$. Otherwise if l senses a free channel then l sends a free message of the form $\langle l, FREE \rangle$.
 - 4: If u is inactive and *asleep* and receives a free message then u becomes *awake*. If u is inactive and decided not to send an I-RTS message in the previous time step and u is *awake* and receives a collision message then u decides to go to *asleep* state with probability $1/2$. If u is inactive and sent an I-RTS in the earlier step and gets an I-CTS then u sends the packet to $\pi(u)$.
-

can treat the actions of active nodes independent of each other. In the following, we can therefore concentrate on a specific active node, say ℓ .

Consider any inactive node-active node pair, (v, ℓ) . Node v has at most $O(\Delta_m)$ inactive nodes in its interference range sending an I-RTS message to some leader node. If more than one node is in awake state, with respect to ℓ , decides to send an I-RTS message, then ℓ will send a COLLIDE message. Since the COLLIDE message will be received by the inactive nodes within r_t range of ℓ awake nodes that decided not to send a CLIENT message to ℓ in the previous slot will go to asleep state.

Consider time to be partitioned into groups of consecutive rounds such that each

group ends with a round where the active node ℓ sends either a COLLIDE message or an I-CTS message. A group ending with a COLLIDE message signifies a failed group and a group ending with an I-CTS message is a successful group. Notice that at the end of every group, whether successful or not, all the inactive nodes within the r_t range of ℓ go to awake state (by step 3 of the protocol).

It is not difficult to show that the expected number of rounds in each group, successful or failed, is $O(\log n)$ and any group is successful with constant probability. Due to symmetry reasons any inactive node is equally likely to be send an I-RTS message in a successful group. Thus, during any successful group, for a given pair (v, ℓ) ,

$$\Pr[v \text{ sends an I-RTS message successfully to } \ell] \geq 1/c\Delta_m$$

for some constant $c > 1$.

There are Δ_m inactive nodes that compete to send a packet, and due to symmetry reasons, it then follows that there is a constant probability that in $O(\Delta_m \log n)$ time steps any particular inactive node can send the packet to the corresponding active node. Using Chernoff bounds the stated claim follows.

□

For the queue size, node $u \in U \cup \mathcal{G}$ can associate slots in the queue with distinct nodes in $U \cup \mathcal{G}$ that are its neighbors in $T(s)$. Since there are only $O(d_1)$ such neighbors and nodes have a constant probability of forwarding a packet during each time step, it suffices to have a queue of size $O(d_1^2)$ so that w.h.p. all packets in transit can be accommodated.

We can show the following time bound for any packet to reach s .

Lemma 2.3.8 *Given a connected dominating set of active nodes with constant density at most d_1 in G_{r_t} and a gathering tree $T(s)$ with sink node s , the information gathering algorithm presented above with $p = 1/d_1$ needs at most $O(m + \Delta_m \log^2 n + D(s) + \log n)$ time steps w.h.p so that all the m packets reach the sink s .*

Proof. We use a technique that is similar to the proof of Theorem 2.3.3. Also, once a packet that is at an inactive node reaches an active node, then it stays in the queue of nodes in $U \cup \mathcal{G}$ only before it reaches s . Thus we can consider the time required for any packet as the sum of the time required to reach a nearest active node and the time required to reach s through nodes in $U \cup \mathcal{G}$. The former is bounded by Lemma 2.3.7 for those packets that start from an inactive node. Thus it is left to bound the latter.

For the latter, we proceed as follows. Using the gathering tree, every packet that has to reach s has to travel a distance of at most $5D(s)$. Let the last packet P to reach s arrive at time T_0 and let $u_0 = s$. The packet P then must have reached some node in $u_1 \in U \cup G''$ at time T_1 and is still at node u_1 during all the time steps between T_1 and $T_0 - 1$. Continuing the history of P further, it must have reached some node $u_2 \in U \cup G''$ at time T_2 and during the time steps between T_2 and $T_1 - 1$ is at node u_2 . We can similarly follow the history of P till it reached a node in $U \cup G''$ for the first time. (It could be that the packet is initially at some node in $U \cup G''$ or it has reached a node in $U \cup G''$ from an inactive node.) This means that all the attempts to forward the packet from u_i to u_{i-1} in the above sequence failed between time steps T_i and $T_{i-1} - 1$.

Since the active nodes use reserved time slots, the transmissions of active nodes cannot fail. For a gateway node g , the probability that a transmission fails is at most:

$$\begin{aligned}
& \Pr[g \text{ did not decide to transmit the message}] + \Pr[g \text{ transmitted the message but failed}] \\
& \leq (1 - p) + p(1 - (1 - p)^{d_1 - 1}) = (1 - p) + p(1 - (1 - 1/d_1)^{d_1 - 1}) \leq 1 - p + p(1 - 1/e) \\
& = 1 - p/e,
\end{aligned}$$

for the chosen value of $p = 1/d_1$. Now assume that it took more than $S + \delta$ time steps for the last packet to reach s for some value of $\delta = bS$ with $S = 5d_1 D(s) + m$. Out of these, at most S attempts must have been successful and all the δ attempts have failed.

The probability of this happening is at most:

$$\binom{S + \delta}{\delta} (1 - (p/e))^\delta \leq \left(\frac{e(S + \delta)}{S} \right)^S \cdot (1 - (p/e))^\delta \leq (e(1 + b)e^{-pb/e})^S$$

The above probability is polynomially small if $b = \Omega((1 + \log n/S)1/p)$. Hence, for the packets to reach s , it takes at most $O(D(s) + m + \log n)$ time steps once the packets have reached some node in $U \cup G''$.

Now add both the time bounds, we get the final bound as stated.

□

Finally, ignoring the work performed while sending the RTS/CTS message and the I-RTS/I-CTS message, since we use paths that are only a constant factor away from the shortest paths to s , we get the following result with respect to work.

Theorem 2.3.9 *Once a stable gathering tree has been constructed, the gathering protocol described above needs $O(W'(s, m))$ work to send all the m messages to the sink node s where $W'(s, m)$ denotes the optimal work required to send the m packets to the sink node s .*

Proof. Ignoring the work done to send I-RTS message, it holds that any inactive node u performs optimal work to forward a packet to $\pi(u)$.

For nodes $U \cup G''$ we proceed as follows. We ignore the work done to send/receive RTS/CTS message. Each active node has a fixed time slot, and thus one transmission from each active node is enough to forward the message from the active node ℓ to $\pi(\ell)$ in the gathering tree. It remains to show that the work necessary for the gateway nodes to forward the message (from gateway nodes) in the gathering tree. Note that an active or gateway node only sends a CTS message if a message can be sent to it without interfering with other transmissions. Since every active or gateway node sends a CTS at most once, the number of message transmissions necessary for all active or gateway nodes to receive the message is optimal.

Thus, there are no unsuccessful packet transmissions in the network when using the gathering protocol. Since the paths to the sink node s in the gathering tree are only 5 times longer than the shortest paths, any packet is forwarded along a path of length that is a constant factor away from the shortest paths to s .

Hence, it holds that the gathering protocol is work-optimal, up to constant factors, to send all the packets to the sink node s .

□

2.3.3.3. SELF-STABILIZATION

The algorithms presented for both the stages can be made self-stabilizing as follows. For stage 1, we require that nodes $v \in U$ keep sending the current value of $d'(s, v)$ as ROUTE

messages during the time slot owned by v . When a node w just wakes up, it first sets $d'(s, w) = -\infty$ and $\pi(w) = \text{NULL}$ and listens to the channel for a period of cd_1 time steps for a constant $c \geq 3$. During this period, since other active nodes send their current distance estimate, node w can update its distance estimate and parent pointer accordingly. If additionally $w \in U \cup \mathcal{G}$ then w can participate in sending ROUTE messages after the initial listening period. This would also imply that also stage 2 can recover from any initial configuration if every node that has a packet and continues to retain the packet in case the predecessor pointer is undefined, (NULL). Ignoring those packets in transit that were lost by nodes that have moved or been powered off, all the other packets can indeed reach the sink node once the system recovers till stage 1.

2.4. Conclusion

In this chapter, we presented a more general communication model for wireless ad-hoc networks, and we present a locally self-stabilizing algorithm to construct an overlay network. Our communication model is much more general than previous models. It allows the path loss of transmissions to significantly deviate from the idealistic unit disk model and does not even require the path loss to form a metric. Also, our model is apparently the first proposed for algorithm design that does not only model transmission and interference issues but also aims at providing a realistic model for physical carrier sensing. Based on this model, we propose a local-control protocol for constructing an overlay network. Our algorithm is an extension of Luby's maximal independent set algorithm [Luby, 1985]. We theoretically proved that construction of overlay network takes polylogarithmic number

of communication rounds, with high probability. Furthermore, we designed algorithms for broadcasting and information gathering in wireless ad-hoc networks with using our overlay network construction algorithm.

CHAPTER 3

CONSTRUCTION OF LINE-BASED P2P OVERLAY NETWORKS

Due to their many applications, peer-to-peer systems have recently received a lot of attention both inside and outside of the research community. Peer-to-peer systems are usually dynamic in nature, that is, peers continuously enter and leave the system. Also, peers may fail. In order to handle such a situation, distributed algorithms are needed that can quickly and reliably recover a peer-to-peer system from any inconsistent state. Systems with such a property are also known to be self-stabilizing. In general, self-stabilization of a system guarantees that regardless of a current system's state, the system will converge to a legal state in a finite number of steps. Before presenting previous work on self-stabilization, we start with some basic definitions.

Definition 3.0.1 *A system is a pair $S = (C, \rightarrow)$ where C is a set of states of the system and $\rightarrow$ is a binary transition relation on C . A computation of S is a non-empty sequence $(c_1, c_2, \dots)$ such that for all $i \geq 0$, $c_i \in C$ and $c_i \rightarrow c_{i+1}$.*

In distributed systems the above mentioned c_i expresses a global state, which is the concatenation of the local states of every process and the contents of every communication channel. A commonly used definition of self-stabilization is:

Definition 3.0.2 *A system S is self-stabilizing with respect to a set of legal states $L \subset C$ if for any state $c \in C$, any computation of the system starting with c will eventually reach a state in L .*

The subset L is problem-specific. Often, it is assumed that L is *closed*, i.e. every non-faulty move from a state in L ends in a state in L . An algorithm is said to be

self-stabilizing if it promotes self-stabilization of the system. A *locally self-stabilizing* algorithm is a self-stabilizing algorithm that runs locally at each node and that only assumes *local* knowledge of the network by the nodes (e.g., the algorithm does not assume that nodes know or have an estimate on the number of nodes in the network).

It turns out that the crux of designing truly self-stabilizing algorithms for many structured overlay peer-to-peer networks such as Chord and skip graphs is to find a self-stabilizing protocol that can convert an arbitrary connected graph into a sorted list or ring. For scalability, such algorithms should converge to a sorted list in at most a polylogarithmic number of rounds. A distributed algorithm for arranging the nodes of an arbitrary connected graph into a sorted list in a polylogarithmic number of rounds and with polylogarithmic work for each node is already known [Angluin et al., 2005], but the algorithm is not locally self-stabilizing. In fact, under certain conditions it can run into a deadlock (i.e., the nodes cannot escape a certain invalid state). Hence, there exists the need for developing truly (locally) self-stabilizing algorithms that can converge to a sorted list from an arbitrary initial state in polylogarithmic time.

We are now ready to formally state the problem we consider in this chapter. We are given an arbitrary connected graph $G(V, E)$, $|V| = n$, with node labels which induce a total order on V . For nodes u and v , we say that $u < v$ (resp., $u > v$), whenever the label of node u is smaller (resp., greater) than that of node v . For simplicity, we assume that all the node labels are distinct. We consider the problem of designing a distributed, locally self-stabilizing algorithm that converts G into a sorted list $H(\{v_1, \dots, v_n\}, E_H)$, where $v_i \in V$, and $v_{i-1} < v_i$ and $\{v_{i-1}, v_i\} \in E_H$ for all i .

The idea of self-stabilization in distributed computing first appeared in a classical paper by E.W. Dijkstra in 1974 [Dijkstra, 1974] in which he looked at the problem of self-stabilization in a token ring. Since Dijkstra's paper, self-stabilization has been studied in many contexts, including communication protocols, graph theory problems, termination detection, clock synchronization, and fault containment. For a recent survey see, e.g., [Brzezinski and Szychowiak, 2000], and a *very* comprehensive list of papers can be found in Herman's Self-Stabilization Bibliography from 2002 [Herman, 2002]. For more information about self-stabilizing algorithms we also recommend the book by Dolev [Dolev, 2000].

Self-stabilization issues have also been considered for peer-to-peer networks. In the technical report of the Chord system [Stoica et al., 2001], for example, several techniques such as the weak stabilization protocol and the strong stabilization protocol are presented that allow the Chord network to recover quickly from various kinds of degenerate states which the authors call *pseudo-trees* or *loopy states*. However, the running time of these algorithms depends on the degree of distortion of the degenerate state, and can be as high as linear on the number of nodes. Moreover, no technique is given that allows the Chord network to recover from an arbitrary connected state. Instead, to prevent Chord from ever running into a state outside of pseudo-trees or loopy states, it is suggested that each node keeps $\Theta(\log n)$ successor pointers instead of just one. In this way, nodes can recover their successor pointers with high probability even if the nodes in the system fail with constant probability. Though this protects against faulty nodes, it is known that this technique will not protect against adversarial nodes [Awerbuch and Scheideler, 2004], and therefore it is

still important to understand how to recover from an arbitrary state.

In a technical report on skip graphs [Aspnes and Shah, 2003] the authors propose a self-stabilization protocol in which each node v periodically checks six locally checkable conditions. If one of these conditions is violated, it is locally repaired. It is shown that an overlay network forms a skip graph if and only if all of the six conditions are satisfied for all nodes. However, the local checking of the conditions does not work for arbitrary states but only for states in which the nodes form a degenerate version of a skip graph that may be created due to node faults or inconsistencies in concurrent join or leave operations. As for the Chord repair mechanisms, the total running time for repairing the conditions at each node depends on the distortion of the degenerate state and can be as high as linear.

Self-stabilizing protocols for organizing the nodes into a sorted ring have already been proposed in [Cramer and Fuhrmann, 2005, Shaker and Reeves, 2005]. In the Iterative Successor Pointer Rewiring Protocol (ISPRP) proposed in [Cramer and Fuhrmann, 2005], each node aims at maintaining correct successor pointers. If there is a local inconsistency, i.e., there are two nodes u and v with successor pointers to a node w , w asks either u or v (depending on their label) to repair its successor pointer. In the Ring Network (RN) protocol proposed in [Shaker and Reeves, 2005], each node v periodically initiates the search for a closest successor by sending a search request to a randomly chosen neighbor. Any node w receiving such a request forwards it to the neighbor of w closest to v . This is continued until there is no such neighbor, or a neighbor is found that is closer to v than its closest successor. In the latter case, v is informed about it. Both protocols can have a very long runtime. Hence, the question remains whether there are simple self-stabilizing protocols

with a polylogarithmic runtime.

3.1. Our contributions

We propose linearization as the basic technique to sort any connected graph. This technique works as follows: each node v repeatedly does the following with its neighbors:

- for its left (i.e., smaller) neighbors $u_1, \dots, u_k$ in the order of decreasing labels, v replaces $\{v, u_1\}, \dots, \{v, u_k\}$ by $\{v, u_1\}, \{u_1, u_2\}, \dots, \{u_{k-1}, u_k\}$, and
- for its right (i.e., larger) neighbors $w_1, \dots, w_\ell$ in the order of increasing labels, v replaces $\{v, w_1\}, \dots, \{v, w_\ell\}$ by $\{v, w_1\}, \{w_1, w_2\}, \dots, \{w_{\ell-1}, w_\ell\}$.

As shown in this chapter, the linearization technique is self-stabilizing and transforms any connected graph into a sorted list, but there are graphs for which this can take a long time. Even for random graphs, the pure linearization (PL) technique appears to perform poorly. Our goal is to find self-stabilizing protocols that are guaranteed to have a polylogarithmic runtime (with high probability), or at least a polylogarithmic runtime for random graphs. Hence, we propose several extensions of the linearization technique and experimentally evaluate their performance. These are:

- Linearization with memory: Each node only linearizes the edges that it got in the previous round but also maintains (“remembers”) edges to all nodes that it was connected to at any time. It uses this node list to accelerate the linearization process of the new edges. See Section 3.3 for details.
- Linearization with shortcut neighbors: Each node only linearizes the edges that it

got in the previous round but also maintains edges to a certain set of $O(\log n)$ other shortcut nodes that it learned about over the time. Also, every node exchanges its set of shortcut nodes with every node it is connected with by a new edge. See Section 3.4 for details.

Though linearization with memory (LM) still has a linear runtime in the worst case, our experiments demonstrate that for random graphs the LM protocol significantly outperforms the PL protocol. LM appears to have a polylogarithmic runtime for these graphs, and the node degree seems to be polylogarithmic as well. Since linearization with shortcut neighbors (LSN) is based on the concept of a virtual space, also this protocol can perform poorly in the worst case (extremely unevenly distributed node labels), but our experiments with LSN demonstrate that it is still much better than the LM protocol for random graphs. Thus, LSN appears to be a good candidate for further, rigorous investigations towards the goal of finding a self-stabilizing protocol with polylogarithmic runtime.

In Section 3.2 we present and analyze the pure linearization technique. Afterwards, we study the linearization with memory technique in Section 3.3 and the linearization with shortcut neighbors technique in Section 3.4. Section 3.5 presents and interprets the simulation results for all three techniques. This chapter ends with conclusions.

3.2. Pure Linearization

In this section we present the pure linearization technique. We are given as input a connected undirected graph $G(V, E)$ with node labels. At any time t during the execution of the algorithm, we will let $G(V, E)$ denote the current configuration of graph G , after edges

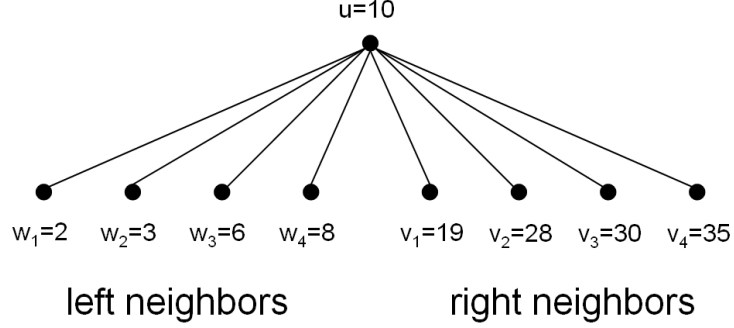


Figure 4. Left and right neighbors of node u

may have been inserted or removed from the original edge set during prior time steps. We start with some definitions:

Definition 3.2.1 *At any time t , node v is a left (resp., right) neighbor of node u if $\{u, v\} \in E$ at time t and $v < u$ (resp., $u < v$).*

Figure 4 illustrates the left and right neighbors of a node u with label 10.

In the pure linearization algorithm (PL), each node repeatedly converts its left neighborhood and its right neighborhood into sorted lists at each round. More specifically, if a node u has left neighbors $w_1 < w_2 < \dots < w_k$ at the start of a left linearization step, it converts its left neighbors into a sorted list by removing the edges $\{u, w_i\}$, $1 \leq i \leq k - 1$ and adding the edges $\{w_i, w_{i+1}\}$, $1 \leq i \leq k - 1$ (Figure 5(a)). Similarly, if $v_1 < v_2 < \dots < v_j$ are the right neighbors of u at the start of a right linearization step, u removes the edges $\{u, v_i\}$, $2 \leq i \leq j$ and adds the edges $\{v_i, v_{i+1}\}$, $1 \leq i \leq j - 1$ (Figure 5(b)) to the graph. Note that in a round every node first converts its left neighbors into a sorted list and then converts its right neighbors into a sorted list. If at any step there is a conflict where a node u wants to remove an edge $\{u, v\}$ and another node w wants to add this edge $\{u, v\}$, the

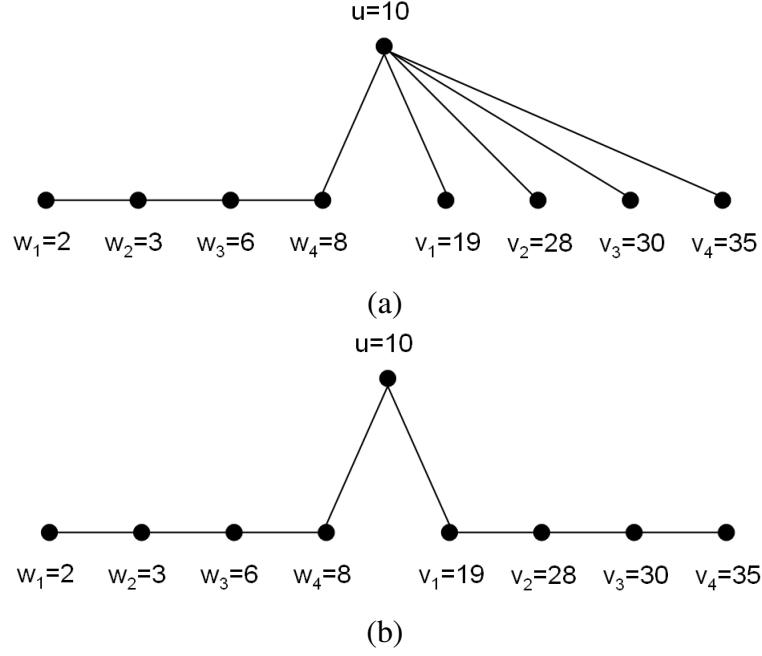


Figure 5. Neighbors of node u after one round of pure linearization.

final outcome will be that the edge $\{u, v\}$ remains in the graph. Algorithm 8 summarizes the PL algorithm.

Algorithm 8 Pure Linearization (PL)

Each node u executes the following:

- 1: **for** each round **do**
 - 2: (Left linearization step) If $w_1 < w_2 < \dots < w_k$ are the current left neighbors of u , u removes the edges $\{u, w_i\}$, $1 \leq i \leq k - 1$ and adds the edges $\{w_i, w_{i+1}\}$, $1 \leq i \leq k - 1$ to the graph.
 - 3: (Right linearization step) If $v_1 < v_2 < \dots < v_j$ are the current right neighbors of u , u removes the edges $\{u, v_i\}$, $2 \leq i \leq j$ and adds the edges $\{v_i, v_{i+1}\}$, $1 \leq i \leq j - 1$ to the graph.
 - 4: **end for**
-

The following two sections provide a formal analysis of the complexity and several other properties of the PL.

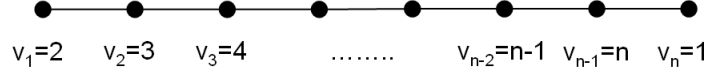


Figure 6. Lower bound example for PL algorithm

3.2.1. Analysis.

Note that the PL algorithm is memoryless, that is, it only needs to know the current state of the connections in order to converge to a sorted list. Hence, if we can show that the PL algorithm converges to a sorted list for any connected graph, the PL algorithm is self-stabilizing. In the following two theorems we will prove matching lower and upper bounds showing that in the worst case the PL algorithm requires $(n - 2)$ rounds to converge to a sorted list.

Theorem 3.2.2 (Lower bound) *The pure linearization algorithm requires at least $(n - 2)$ rounds to generate a sorted list out of an arbitrary graph G .*

Proof. Consider the example graph $G(V, E)$, $E = \{\{i, i + 1\} | i \in \{2, 3, \dots, n - 1\}\} \cup \{\{n, 1\}\}$ illustrated in Figure 6. After the first round, the edge $\{n, 1\}$ is removed and the edge $\{n - 1, 1\}$ is added to G . After the second round, the edge $\{n - 1, 1\}$ is removed and the edge $\{n - 2, 1\}$ is added to G . After the third round, the edge $\{n - 2, 1\}$ is removed and the edge $\{n - 3, 1\}$ is added to G . The PL algorithm will take $(n - 2)$ rounds to convert G to a sorted list since after the j^{th} round, the edge $\{n - j + 1, 1\}$ is removed and the edge $\{n - j, 1\}$ is added to G , $1 \leq j \leq n - 2$.

□

Theorem 3.2.3 (Upper bound) *After $(n-2)$ rounds, the pure linearization algorithm will generate a sorted list out of any graph G .*

Proof. Let $V = \{a_1, a_2, a_3, \dots, a_n\}$, where $a_i < a_j$ for all $i < j$, $i, j \in \{1, 2, \dots, n\}$.

Recall that one round of the PL algorithm consists of a left linearization step followed by a right linearization step. The following three claims will be used in the proof of Theorem 3.2.3. The claims bound the number of left and right neighbors out of each node after each round of the PL algorithm.

Claim 3.2.4 *After k rounds $a_n, a_{n-1}, a_{n-2}, \dots, a_{n-k+1}$ will each have exactly one left neighbor for every $k = 1, 2, \dots, n-2$.*

Proof. We will prove this claim by induction on k .

Base case: $k = 1$. We will show that after the first round a_n will have exactly one left neighbor. During the left linearization step of the first round, a_n will remove all of its left neighbors except for one. Since a_n is the largest node, it does not have any right neighbors. Thus no node can add a left neighbor to a_n during a left linearization step. Hence after the left linearization step, a_n will have only one left neighbor, which we call b_n . If b_n only has one right neighbor (a_n itself) after the right linearization step, the left neighbor of a_n will not change. If b_n has more than one right neighbor, after the right linearization step, the left neighbor of a_n will change and it will be the second largest right neighbor of b_n . Thus, after the first round a_n will have only one left neighbor.

Inductive Hypothesis: After k rounds $a_n, a_{n-1}, a_{n-2}, \dots, a_{n-k+1}$ will each have exactly one left neighbor, $k < n-2$.

Inductive Step: We will show that after $k + 1$ rounds $a_n, a_{n-1}, a_{n-2}, \dots, a_{n-k+1}, a_{n-k}$ will each have exactly one left neighbor, $k \leq n - 2$.

We will first show that after $k + 1$ rounds $a_n, a_{n-1}, a_{n-2}, \dots, a_{n-k+1}$ will still have one left neighbor each. From the inductive hypothesis, we know that after k rounds $a_n, a_{n-1}, a_{n-2}, \dots, a_{n-k+1}$ will have only one left neighbor each. Consider node a_i , $n - k + 1 \leq i \leq n$ at round $k + 1$. Since a_i only has one left neighbor it will not do anything during the left linearization step of this round. Since all nodes $a_j > a_i$ have only one left neighbor each, they cannot add a left neighbor to a_i . So, after the left linearization step of round $(k + 1)$, a_i will still have only one left neighbor (say b_i). If b_i has no right neighbor smaller than a_i , the left neighbor of a_i will not change after right linearization step of round $k + 1$. If b_i has a right neighbor smaller than a_i , the left neighbor of a_i will change and it will be the largest right neighbor of b_i which is less than a_i . Thus, after $k + 1$ rounds a_i will have only one left neighbor, $n - k + 1 \leq i \leq n - 1$.

Secondly, we will show that after $k + 1$ rounds a_{n-k} will also have only one left neighbor. From the inductive hypothesis, we know that after k rounds $a_n, a_{n-1}, a_{n-2}, \dots, a_{n-k+1}$ will have only one left neighbor each. We start by showing that a_{n-k} must have at least one left neighbor after round $k + 1$. Then we will show that a_{n-k} cannot have more than one left neighbor after this round. By contradiction, assume that after k rounds a_{n-k} has no left neighbor. Since all nodes greater than a_{n-k} have only one left neighbor each, all of the right neighbors of a_{n-k} must have only one left neighbor

which is a_{n-k} (if a_{n-k} has no right neighbor the graph is disconnected.). Since the graph is connected, a_{n-k} must have a path to a_1 . However, the right neighbors of a_{n-k} cannot have a left neighbor smaller than a_{n-k} (since a_{n-k} is their only left neighbor) and similarly the right neighbors of the right neighbors of a_{n-k} cannot have a left neighbor smaller than a_{n-k} , and so on. Therefore, there is no path between a_{n-k} and a_1 which is a contradiction.

Now we show that a_{n-k} cannot have more than one left neighbor after round $(k+1)$. We have just seen that after k rounds a_{n-k} has at least one left neighbor. During the left linearization step of round $(k+1)$, a_{n-k} will remove all of its left neighbors except one. Since all nodes $a_j > a_{n-k}$ have only one left neighbor, they cannot add a left neighbor to a_{n-k} . So, after the left linearization step, a_{n-k} will have only one left neighbor (say b_{n-k}). If b_{n-k} has no right neighbor smaller than a_{n-k} , the left neighbor of a_{n-k} will not change during the right linearization step of round $(k+1)$. If b_{n-k} has a right neighbor smaller than a_{n-k} , the left neighbor of a_{n-k} will change and it will become the largest right neighbor of b_{n-k} which is less than a_{n-k} . Hence, after $k+1$ rounds a_{n-k} will have exactly one left neighbor, completing the proof.

□

Claim 3.2.5 *After $n-2$ rounds all nodes except a_1 will have exactly one left neighbor each.*

Proof. From Claim 3.2.4 we know that after $n-2$ rounds $a_n, a_{n-1}, a_{n-2}, \dots, a_3$ will have one left neighbor each. We show that a_2 will also have exactly one left neighbor (a_1). After $n-2$ rounds, since the graph is connected, there must be a path between a_2 and

a_1 . By contradiction, assume a_2 and a_1 are not neighbors, implying that a_2 has only right neighbors. By Claim 3.2.4, the right neighbors of a_2 only have right neighbors and so on. Thus we cannot reach a_1 from a_2 , a contradiction.

□

Claim 3.2.6 *After $n - 2$ rounds all nodes except a_n will have exactly one right neighbor each.*

Proof. This statement is the dual of Claim 3.2.5 and its proof is analogous. The main difference in the proof is that we start the algorithm with a left linearization, but this will not affect the proof significantly.

□

We are now ready to prove Theorem 3.2.3. We know, by Claim 3.2.5 and Claim 3.2.6, that each node a_i has exactly one left and one right neighbor after $n - 2$ rounds of the PL algorithm. In addition, we will show that the (only) left neighbor of node a_i must be a_{i-1} , $1 < i \leq n$. This is trivially true for $i = 2$. Assume that the left neighbor of a_i is a_{i-1} , and hence the (only) right neighbor of a_{i-1} is a_i , for all $1 \leq i \leq k$. This trivially implies that the left neighbor of a_{k+1} must be a_k (and the right neighbor of a_k must be a_{k+1}), and hence $a_1, \dots, a_n$ is a sorted list of V .

□

3.2.2. Other Properties of PL

In the following claims, we will present several properties of PL algorithm regarding the degree of the nodes and the required number of rounds.

Claim 3.2.7 *At each round of the PL algorithm, the degree of a node v increases by at most an additive constant.*

Proof. Assume a node u has l left and r right neighbors at the start of a round. After the left linearization step of this round, node u will have (i) at most $r+1$ left neighbors, since each of its right neighbors can add at most one more left edge to u and only one of its previous left neighbors will remain; and (ii) node u will have at most r right neighbors, since each of its right neighbors will either remain or be changed to another node. In summary, after the left linearization step, node u will have at most $r + 1$ left and r right neighbors. Similarly, after the right linearization step of this round, node u will have at most $r + 1$ left and $r + 2$ right neighbors. Following the same argument, after the left linearization step of the next round, node u will have at most $r + 3$ left and $r + 2$ right neighbors, and after the corresponding right linearization step, node u will have $r + 3$ left and $r + 4$ right neighbors. Hence, after t left and right linearizations steps, node u will have at most $r + 2t - 1$ left and $r + 2t$ right neighbors.

□

An edge $\{u, w\}$ is said to *cross* a node v if $u \leq v \leq w$. The following claim bounds the degree of a node v by the number of edges crossing the node.

Claim 3.2.8 *The maximum degree of a node v during the PL algorithm is bounded by the number of edges crossing it.*

Proof. Linearization ensures that for any edge (u, w) converted into (u', w') during a linearization step: $u \leq u' \leq w' \leq w$.

□

3.3. Linearization with Memory

In this section we present an extension of the pure linearization technique called the linearization with memory (LM). In the linearization with memory algorithm, nodes remember all of their old neighbors. While placing current neighbors into sorted lists, a node u will also use its old neighbors in order to "better place" its current neighbors within the final total order, as we will show.

In the LM algorithm, nodes will place their current left and right neighbors into sorted lists by reconnecting them. When placing a left neighbor v into the sorted list, a node u will remove the edge $\{u, v\}$ and edge $\{v, w\}$ will be added, where w is the smallest node u remembers (w may be a current neighbor of node u) such that $v < w$. If v is the largest left neighbor of node u , edge $\{u, v\}$ remains. When placing a right neighbor v into the sorted list, a node u will remove the edge $\{u, v\}$ and edge $\{w, v\}$ will be added, where w is the largest node u remembers (w may be a current neighbor of node u) such that $w < v$. If v is the smallest right neighbor of node u , edge $\{u, v\}$ remains. For example, consider a node u that currently has left neighbors $v_1 < v_2 < v_3 < v_4$ and in addition remembers the nodes $w_1 < w_2 < w_3 < w_4$, where $w_1 < v_1 < v_2 < w_2 < v_3 < w_3 < w_4 < v_4 < u$. The LM algorithm would place u 's left neighbors into the sorted list by removing the edges $\{u, v_1\}, \{u, v_2\}, \{u, v_3\}$ and adding the edges $\{v_1, v_2\}, \{v_2, w_2\}, \{v_3, w_3\}$, as illustrated in Figure 7. Algorithm 9 summarizes the LM technique.

While the LM technique alone still has a linear worst-case running time (see Corol-

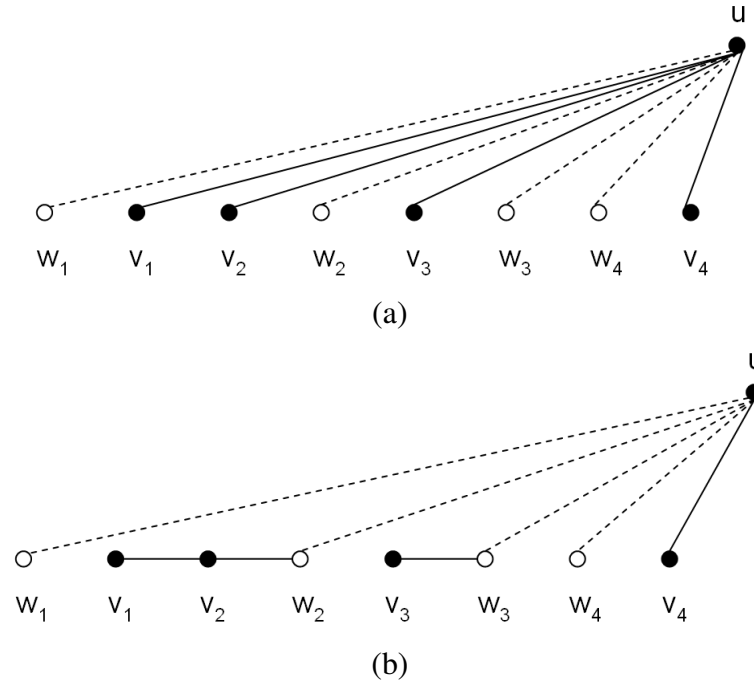


Figure 7. Node u places its current left neighbors into sorted list by reconnecting them.

Algorithm 9 Linearization with Memory (LM)

Each node u executes the following:

- 1: **for** each round **do**
 - 2: Node u will remember all of its neighbors
 - 3: For each current left neighbor v of u , u replaces edge $\{u, v\}$ by edge $\{v, w\}$, where w is the smallest node u remembers such that $v < w$, if such a node w exists. Otherwise, u keeps the edge $\{u, v\}$.
 - 4: For each current right neighbor v of u , u replaces edge $\{u, v\}$ by edge $\{w, v\}$, where w is the largest node u remembers such that $w < v$, if such a node w exists. Otherwise, u keeps the edge $\{u, v\}$.
 - 5: **end for**
-

lary 3.3.1), its average running time seems to be polylogarithmic as the experimental results in Section 3.5 show, improving significantly on the average running time of the PL algorithm. The major drawback of the LM technique is that the degree of a node may increase very fast, since nodes "never forget" a neighbor (see Table 2). The linearization with shortcut neighbors (LSN), introduced in Section 3.4, is a refinement of the LM technique that aims at keeping the degree of the nodes lower (and at speeding up the LM process).

We now show that LM algorithm requires $n - 2$ rounds to generate a sorted list of the nodes in the worst case. For any two nodes v and w , where $u < w$, let $|w - u|$ denote the cardinality of the set $\{v : u < v \leq w\}$.

Theorem 3.3.1 *The linearization with memory algorithm still requires $n - 2$ rounds to generate a sorted list of the nodes in the worst-case.*

Proof. The LM algorithm will behave just like the PL algorithm for the graph in Figure 6 and hence the lower bound of Theorem 3.2.2 also applies to the LM algorithm.

The following two claims are needed to prove that the LM algorithm requires at most $n - 2$ rounds to generate a sorted list of the nodes.

Claim 3.3.2 *At any round of the algorithm, if a node u remembers a node $v < u$ (node v may be a current neighbor of u), then there exists an increasing path between u and v in the current graph, i.e. there exists a path $P = v_1, v_2, \dots, v_p$, where $v_1 = v$, $v_p = u$, and v_i is a current left neighbor of v_{i+1} , $1 \leq i \leq p - 1$ (also implying that $v_1 < v_2 < \dots < v_p$, $p \geq 2$).*

Proof. Let v be a left neighbor that u currently remembers. We will prove by induction on $|u - v|$.

Base case: $|u - v| = 1$. Node v is thus the largest left neighbor node u has seen. According to the LM algorithm, v must still be a current left neighbor of node u , and hence $P = vu$.

Inductive Hypothesis: If $|u - v| \leq k, k \geq 1$, there exists an increasing path between u and v in the current graph.

Inductive Step: We will show that if $|u - v| = k + 1$, then there exists an increasing path P between u and v in the current graph.

If v is a current left neighbor of node u , $P = vu$ and we are done. Thus assume that node u only remembers node v . Since v is not a current left neighbor of node u , either u reconnected node v (i.e., u replaced edge $\{u, v\}$ by another edge in step 3 of the LM algorithm) or v reconnected node u , at some prior time. Without loss of generality, assume that u reconnected node v (the proof is analogous if v reconnected u), by replacing edge $\{u, v\}$ by an edge $\{v, w\}$, where w was (and may still be) a current neighbor of u and $v < w < u$. Hence $|v - w|$ and $|w - u|$ are at most k (and u remembers both v and w currently). By the induction hypothesis, there must exist increasing paths P_1 and P_2 in the current graph between v and w and between w and u , respectively. Hence P_1P_2 is an increasing path between u and v in the current graph and the induction hypothesis follows.

□

Claim 3.3.3 *The graph remains connected after any number of rounds of the LM algorithm.*

Proof. From Claim 3.3.2, we know that if a node u remembers a node v , there exists an increasing path between u and v in the current graph (either v was a left neighbor of u

at some prior time, or u was a left neighbor of v , and the theorem above applies). Nodes will always remember their initial neighbors. So, from Claim 3.3.2, there exists a current neighbor path between nodes and their initial neighbors. So, graph will always remain connected.

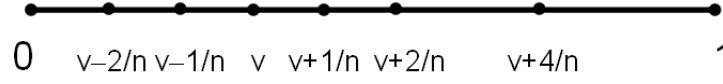
□

Claim 3.3.3 shows that the graph will remain connected. Thus, Claims 3.2.4, 3.2.5, and Claim 3.2.6 all hold for the LM algorithm (taking into account only the current neighbors of a node, and *not* all the neighbors the node remembers). Hence, the upper bound in Theorem 3.2.3 also applies to the LM algorithm.

□

3.4. Linearization with Shortcut Neighbors

In this section we present the linearization with shortcut neighbors (LSN) technique. This technique can be seen as a variation of the LM technique. The main ideas behind the LSN algorithms are that (i) a node v only remembers at most $2(\log n + 1)$ carefully selected shortcut neighbors, which function as representatives of exponentially increasing length intervals out of node v , and (ii) at each round, a node exchanges information about its shortcut neighbors with all of its current or shortcut neighbors at that time. Without loss of generality, we assume that each node has an arbitrary label belonging to the interval $[0, 1)$. Each node v divides $[v, 1)$ into the intervals $I_i^+(v) = [v + 2^{i-1}/n, v + 2^i/n)$, for all $1 \leq i \leq k$ such that $v + 2^k/n < 1$ and $v + 2^{k+1}/n > 1$; and the intervals $I_0^+(v) = [v, v + 1/n)$ and $I_{k+1}^+(v) = [v + 2^k/n, 1)$. Each node v divides $[0, v]$ into the intervals $I_i^-(v) = [v - 2^i/n,$

Figure 8. Intervals for node v

$v - 2^{i-1}/n$), for all $1 \leq i \leq j$ such that $v - 2^j/n > 0$ and $v - 2^{j+1}/n < 0$; and the intervals $I_0^-(v) = [v - 1/n, v]$ and $I_{j+1}^+(v) = [0, v - 2^j/n]$. For each of the intervals I_i^+ and I_i^- , node v will remember one shortcut neighbor that belongs to the interval, if such a node exists. Hence v will remember at most $2(\log n + 1)$ shortcut nodes. Those shortcut neighbors that v remembers will aid v in placing its current neighbors within the sorted list, in the same way as the LM algorithm.

Algorithm 10 Linearization with Shortcut Neighbors (LSN)

Each node u executes the following:

- 1: **for** each round **do**
 - 2: Send your shortcut neighbors to your original and shortcut neighbors
 - 3: Receive shortcut neighbors of your neighbors.
 - 4: Check the shortcut neighbors u just received from its neighbors and the current neighbors u has. If there exists a node v that belongs to an interval $I_i^+(u)$ or $I_i^-(u)$ which was not covered by any of the prior neighbors of node u (i.e., there is no shortcut neighbor for this interval at node u), add v to u 's shortcut neighbors list (if there are two or more such nodes, select one of them arbitrarily).
 - 5: Reconnect the current left neighbors of u , as described in Step 3 of the LM algorithm (node u only remembers its current neighbors and shortcut neighbors, at any time).
 - 6: Reconnect the current right neighbors, as described in Step 4 of the LM algorithm.
 - 7: **end for**
-

The intuition behind the LSN algorithm is our hope that by using geometrically decreasing intervals near each node and collecting neighbors for these, the network topology converges quickly against something close to a hypercubic network. A hypercubic network structure would have the advantage that nearest neighbors can be found much quicker (basically in logarithmic time via binary search) than in arbitrary networks. In fact, it is not

difficult to show that linearization on top of a hypercubic network would result in a sorted line within $O(\log n)$ communication rounds. Algorithm 10 describes the LSN algorithm in detail.

The LSN algorithm also performs poorly in the worst-case, since, like the PL and LM algorithm, it is based on the concept of a virtual space (an extremely unevenly distribution of node labels would hurt the algorithm). However, it is not clear what the *exact* worst-case complexity of the LSN algorithm is. The upper bound given by Theorem 3.2.3 still applies to the LSN algorithm. However, since nodes now exchange their neighbor information with each other, the example used in the proof of Theorem 3.2.2 no longer gives a lower bound of $n - 2$.

3.5. Experimental Results

3.5.1. Regular Graphs

All three algorithms, PL, LM, and LSN, are implemented with Java. The runtimes of the algorithms are first compared for connected random graphs such that the expected degree of each node is 10. The random graphs are generated such that the probability of existence of each edge is $10/(n - 1)$, where n is number of nodes in the graph. If the generated graph is not connected, a new graph is generated. We run experiments for graphs with up to 16000 nodes. Each experiment is repeated 1000 times. We calculate the average number of required rounds.

The experimental results show that the LSN algorithm improves over the LM algorithm and both LM and LSN algorithms seem to have a polylogarithmic runtime (Table 1

	500	1000	2000	4000	8000	16000
PL	122	275.7	634.6	1347	2835	5485.2
LM	26.3	45.41	79.47	135.2	227.7	394.83
LSN	18.5	25.97	35.28	43.93	53.4	65.1

Table 1. Average number of rounds for PL, LM and LSN (for each node avg number of neighbors is 10)

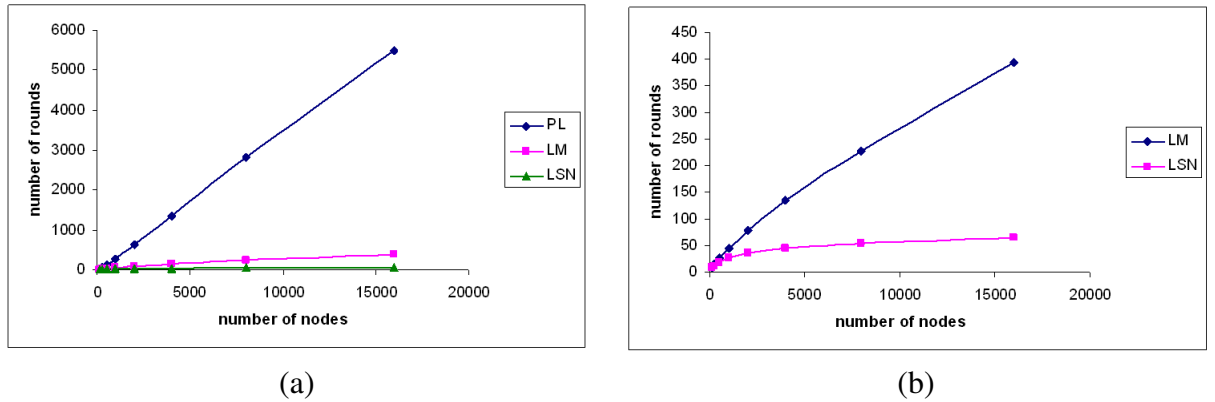


Figure 9. Average number of rounds for PL, LM and LSN

and Figure 9). The former observation is somewhat surprising when taking into account that the nodes in LM may collect much more edges over time than LSN. However, the fact that LSN uses a structured approach of collecting edges still seems to give it an advantage.

Next, for the LM and LSN algorithms the average maximum node degree is computed experimentally (Figure 10 and Table 2). As expected, the LM algorithm has a greater average maximum degree than the LSN algorithm. Still, the experimental results indicate that the maximum node degree might be polylogarithmic for both algorithms.

	50	100	200	400	1000	2000
LM	25.9	35	46.1	58.38	76.8	92.1
LSN	12.9	17.1	22.1	27.15	33.39	37.74

Table 2. Average maximum node degree for LM and LSN (for each node avg number of neighbors is 10)

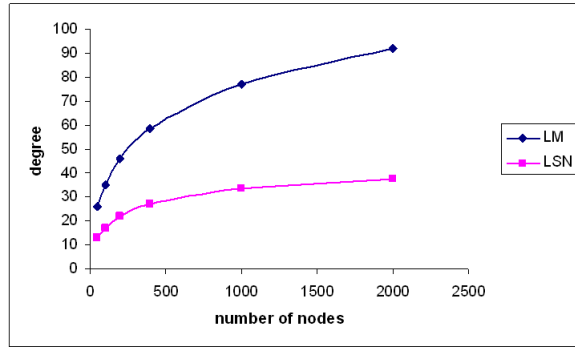


Figure 10. Average maximum node degree for LM and LSN (for each node avg number of neighbors is 10)

3.5.2. Power Law Graphs

In recent work, it is claimed that the Internet and certain unstructured P2P network topologies satisfy the power law [Faloutsos et al., 1999, Broder et al., 2000, Barabasi and Albert, 1999, Ripeanu et al., 2002].

Definition 3.5.1 (Power law) *The number of nodes that have degree k is proportional to $1/k^\alpha$, where α is a constant number and $\alpha > 1$.*

We generated random power law graphs according to the Chung-Lu model [F. Chung, 2002], with $\alpha = 2$ and ran some experiments. To generate these graphs, we first generated the degrees of the nodes according to the power law. Then we considered every node pair $\{i, j\}$ and selected an edge between them with probability $\frac{d_i * d_j}{2m}$ where d_i is the desired degree of node i and m is number of edges in the graph (Chung-Lu model [F. Chung, 2002]). Thus, we have a graph with node i having expected degree d_i .

As for regular random graphs, the experimental results indicate that the LM and LSN algorithms seem both to have an expected polylogarithmic runtime for random power

	500	1000	2000	4000	8000	16000
PL	170	339	699.2	1420	2786	7142
LM	13.9	21.62	30.08	39.74	54.15	62.67
LSN	12.1	15.21	18.12	22.97	28.08	38.167

Table 3. Average number of rounds for PL, LM and LSN (graphs satisfy power law with $\alpha = 2$)

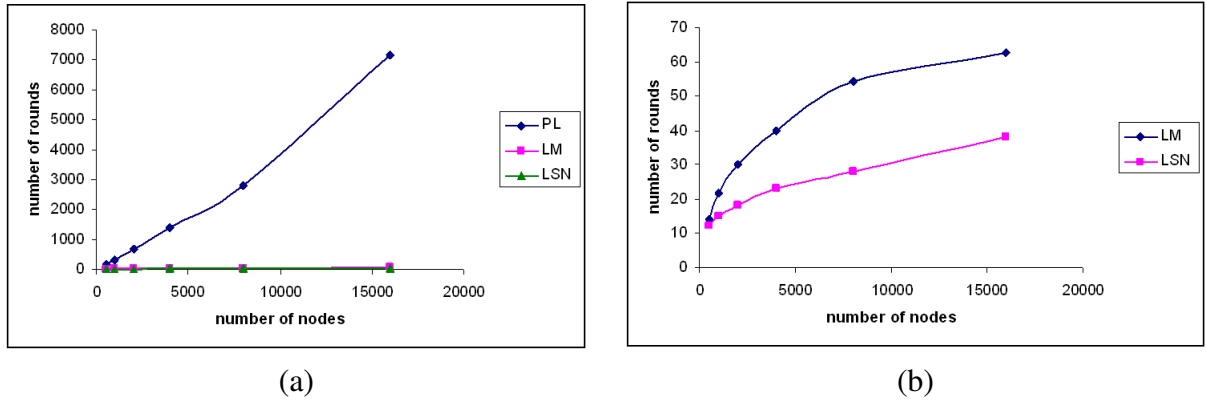


Figure 11. Average number of rounds for PL, LM and LSN (graphs satisfy power law with $\alpha = 2$)

law graphs (Table 3 and Figure 11).

3.5.3. Bounded Locality

We generated random graphs with *bounded locality* B , i.e. each edge in the graph crosses at most B nodes. Recall that an edge $\{u, v\}$ crosses a node w if $u < w < v$. We generated random graphs with 1000 nodes and expected node degree of 10. We ran experiments for different values of B , by letting $B = 125, 250, 500, 1000$.

	125	250	500	1000
PL	45.18	88.25	181.1	275.7
LM	13.82	22	34.13	45.41
LSN	11.04	15.36	21.55	25.97

Table 4. Average number of rounds for PL, LM and LSN with bounded locality

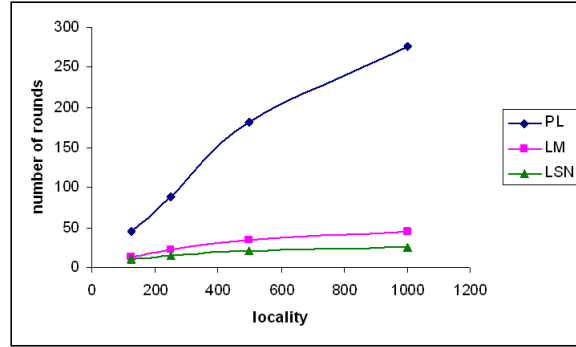


Figure 12. Average number of rounds for PL, LM and LSN with bounded locality

The experimental results also indicate that the expected running time of the PL algorithm increases linearly and that the expected running time of both the LM and LSN algorithms increase polylogarithmically with B (Table 4, Figure 12).

3.6. Conclusion

In this chapter, we presented locally self stabilizing algorithms for constructing peer-to-peer overlay networks [Onus et al., 2007]. The crux of constructing structured P2P overlay networks which has line or ring structure such as Chord or skip graphs is to design an algorithm that can convert an arbitrary connected graph into a sorted list. We presented the *linearization* technique where each node repeatedly converts its neighbors into a sorted list. We proved that the worst-case running time of the PL algorithm is linear. Hence, we further looked at proper extensions of the algorithm, namely, the linearization with memory (LM) and linearization with shortcut neighbors (LSN) algorithms. Both algorithms are based on the idea of remembering old neighbors. Experimental results on random graphs, random power law graphs and random graphs with bounded locality show that the LM and LSN algorithms significantly improve the PL algorithm and that these algorithms seem to have

a polylogarithmic time complexity in expectation.

CHAPTER 4

PUBLISH-SUBSCRIBE OVERLAY NETWORK DESIGN

In the publish/subscribe (pub/sub) communication paradigm, publishers and subscribers interact in a decoupled fashion. Publishers publish their messages through logical channels and subscribers receive the messages they are interested in by subscribing to the appropriate services, which deliver messages through these channels.

A pub/sub system may be *topic-based*, if messages are published to “topics”, where each topic is uniquely associated with a logical channel. Subscribers in a topic-based system will receive all messages published to the topics to which they subscribe. The publisher is responsible for defining the classes of messages to which subscribers can subscribe. In a *content-based* system, messages are only delivered to a subscriber if the attributes of those messages match constraints defined by the subscriber; each logical channel is characterized by a subset of these attributes. The subscriber is responsible for classifying the messages.

Pub/sub communication systems are scalable and simple to implement (see e.g., [Oracle, 2000, Anceaume et al., 2006, Baehni et al., 2004, Baldoni et al., 2007a, Banerjee et al., 2002, Bhola et al., 2002, Carzaniga et al., 2004, Castro et al., 2002, Chand and Felber, 2005, Levis, 1999, Voulgaris et al., 2006, Ramasubramanian et al., 2006, Sandler et al., 2005, Tam et al., 2003, Tock et al., 2005]).

Hence there are many applications which are built on top of such systems, most notably a plethora of Internet-based applications, such as stock-market monitoring engines, RSS [Tock et al., 2005] feeds [Liu et al., 2005], on-line gaming and several others. For a survey on pub/sub systems, see [Eugster et al., 2003].

In this chapter, we will design a (peer-to-peer) overlay network for each pub/sub

topic, in the sense that for each topic t , the subgraph induced by the nodes interested in t will be connected. This translates into a *fully decentralized* topic-based pub/sub system since any given topic-based overlay network will be connected and thus nodes subscribed to a given topic do not need to rely on other nodes (agents) for forwarding their messages. Such an overlay network is called *topic-connected*.

We can evaluate the complexity of a pub/sub overlay network in terms of the cost of topic-based broadcasts on the network. As in many other systems, a space-time trade-off exists: On one hand, one would like the total time taken by the broadcast (which directly depends on the diameter of each topic-based subnetwork) to be as small as possible; on the other hand, for memory and node bandwidth considerations, one would like to keep the total degree of a node small. Those two measures are often conflicting. For example, take the simple scenario where all nodes are subscribed to the same topic: A star overlay would result in the best possible diameter but worst possible degree for the nodes. Even if we were to maintain a balanced structure (e.g., a balanced binary tree) for each topic, it is not clear how to achieve that without letting the node degrees grow as large as the sizes of the node subscription sets.

Some of the current solutions adopted in practice actually fail at maintaining *both* the diameter and the node degrees low. A naive, albeit popular, solution to topic connected-overlay network design is to construct a cycle (or a tree or any other separate overlay structure) connecting all nodes interested in a topic independently for each given topic [Voulgaris et al., 2006]: This construction may result in a network with node degrees proportional to the nodes' subscription sizes, whereas a more careful construction, taking

into account the correlations among the node subscription sets might result in much smaller node degrees (and total number of edges).

Low node degrees are desirable in practice for scalability and also due to bandwidth constraints. Nodes with a high number of adjacent links will have to manage all these links (e.g., monitor the availability of its neighbors, incurring in heartbeats and keep-alive state costs, and connection state costs in TCP) and the traffic going through each of the links, without being able to take great advantage of aggregating the traffic (which would also reduce the number of packet headers, which can be responsible for a significant portion of the traffic for small messages). See [Chockler et al., 2007b] for further motivation.

The node degrees and number of edges required by a topic-connected overlay network will be low if the node subscriptions are well-correlated. In this case, by connecting two nodes with many coincident topics, one can satisfy connectivity of many topics for those two nodes with just one edge. Several recent empirical studies suggest that correlated workloads are indeed common in practice [Tock et al., 2005, Liu et al., 2005].

In Section 4.2, we consider the problem of devising topic-based pub/sub overlay networks with low node degrees. More specifically, we consider the following problem:

Minimum Maximum Degree Topic-Connected Overlay (MinMax-TCO) Problem: Given a collection of nodes V , a set of topics T , and the node interest assignment I , connect the nodes in V into a topic-connected overlay network G which has the least possible maximum degree.

In Section 4.2, our main contribution is the formal design and analysis of the topic-connected overlay design algorithm (MinMax-ODA) which approximates the MinMax-

	Chockler et. al.	Section 4.2	Section 4.3	Lower Bound
Avg Degree	$O(\log(n * t))$	$\theta(n)$	$O(\min\{k * \log(n * t), n\})$	$\Omega(c)$
Max Degree	$\theta(n)$	$O(\log(n * t))$	$O(\min\{(n/k) * \log(n * t), n\})$	$\Omega(c)$

Table 5. Summary of known results on overlay network construction for publish/subscribe communication (n : number of nodes, t : number of topics, c : constant, k is any parameter between 1 and n)

TCO problem within a logarithmic factor. The MinMax-ODA algorithm is a greedy algorithm which relies on repeatedly using a greedy approach for finding matchings that connect a large (close to maximum) number of different connected components which emerge for the given topics. We also show that no polynomial time algorithm can approximate the MinMax-TCO problem within a constant factor (unless $P=NP$), and so our MinMax-ODA algorithm is almost tight. No previous algorithm with sublinear approximation guarantees on the maximum degree of a topic-connected pub/sub overlay network was known prior to this work. Furthermore, we validate the performance of MinMax-ODA with experimental results.

In Section 4.3, we consider the problem of devising topic-based pub/sub overlay networks with low node degrees. One could argue for keeping the maximum degree of a node low or for keeping the overall average node degree low, since both are important and relevant measures of the complexity and scalability of a system. Unfortunately, previous attempts at minimizing either one of these degree measures alone [Chockler et al., 2007a, Onus and Richa, 2009b] resulted in a linear explosion for the other measure (see Table 5). Hence it is only natural to consider the following problem [Onus and Richa, 2009a]:

Low Degree Topic-Connected Overlay (Low-TCO) Problem: Given a collection of nodes

V , a set of topics T , and the node interest assignment I , connect the nodes in V into a topic-connected overlay network G which has low average and maximum degree.

We present a parameterized sublinear approximation algorithm (Low-ODA) for this problem which approximates both the average and the maximum degree well (See Table 5). To the best of our knowledge, this is the first overlay network design algorithm that achieves sublinear approximations on both the average and maximum degrees (e.g., for $k = \sqrt{n}$). The Low-ODA algorithm is a greedy algorithm which relies on repeatedly evaluating the trade-off of greedily adding an edge that would not increase the maximum degree versus greedily adding an edge that would lead to a small number of total edges in the final overlay network. The main contribution of Section 4.3 is therefore to show that such a greedy approach can work and indeed leads to non-trivial sublinear approximation on both the average and maximum degree. We expect that the greedy parameterized template introduced by our algorithm will lead to applications in other network design domains where scalability is a key issue.

In Section 4.4, we propose a variation of the MinMax-TCO problem by enforcing that each topic-connected overlay network be of constant diameter, while keeping the average degree low. We present three heuristics for this problem which guarantees that each topic's induced overlay subnetwork will be of diameter 2 and which aims at keeping the average node degree of the overall topic-connected overlay network low. We validate these algorithms through experimental results.

4.1. Related Work

In [Chockler et al., 2007a], Chockler et al. introduced a closely related problem to the MinMax-TCO problem, which we call *MinAv-TCO* [In the original paper, this problem was called Min-TCO; since it aims at *minimizing the average* degree of the overlay network, and in order to avoid confusion with the MinMax-TCO problem considered in this paper, we will refer to the problem considered by Chockler et al. in [Chockler et al., 2007a] as MinAv-TCO in our work.]. The MinAv-TCO problem aims at minimizing the average degree of the nodes rather than the maximum degree. They present an algorithm, called GM, which achieves a logarithmic approximation on the minimum average degree of the overlay network. While minimizing the average degree is a step forward towards improving the scalability and practicality of the pub/sub system, their algorithm may still produce overlay networks of very uneven node degrees where the maximum degree may be unnecessarily high: As we will show in Section 4.2, their algorithm may produce a network with maximum degree $|V|$ while a topic-connected overlay network of constant degree exists for the same configuration of I . Some of the high level ideas and proof techniques of [Chockler et al., 2007a] have their roots in techniques used for the classical Set-Cover problem. We benefit from some of the ideas in [Chockler et al., 2007a] and also build upon the constructions for Set-Cover, extending and modifying them to be able to handle the maximum degree case.

To the best of our knowledge, minimizing max-degree or diameter in topic-connected pub/sub overlay network design had not been directly addressed prior to this thesis. The overlay networks resulting from [Anceaume et al., 2006, Baldoni et al., 2007b,

Chand and Felber, 2005] are not required to be topic-connected. In [Baldoni et al., 2007a, Castro et al., 2002, Chockler et al., 2007b, Voulgaris et al., 2006], topic-connected overlay networks are constructed, but they make no attempt to minimize the average or maximum node degree. The first to directly consider node degrees when building topic-connected pub/sub systems were Chockler et al. in [Chockler et al., 2007a], as we mentioned above.

The importance of minimizing both maximum and average degree has been recognized in some network domains, such as that of survivable network design [Lau et al., 2007] and that of establishing connectivity in wireless networks [Santis et al., 2007]. To the best of our knowledge, minimizing both the maximum and the average degree in topic-connected pub/sub overlay network design had not been directly addressed prior to this thesis.

4.2. Minimum Maximum Degree Publish-Subscribe Overlay Network Design

This section is organized as follows. In Section 4.2.1, we present some definitions and restate the formal problem definition. In Section 4.2.2, we present an outline of the related problem of minimizing the average node degree, namely the MinAv-TCO problem, and the corresponding logarithmic approximation algorithm GM proposed by Chockler et al. [Chockler et al., 2007a], since some of the ideas presented will be useful for the MinMax-TCO problem. Section 4.2.3 presents our topic-connected overlay design algorithm MinMax-ODA, whose approximation ratio is proved in Section 4.2.4; in Section 4.2.5, we present the hardness of approximation results for MinMax-TCO.

4.2.1. Preliminaries

Let V be the set of nodes, and T be the set of topics. Let $n = |V|$. The interest function I is defined as $I : V \times T \rightarrow \{0, 1\}$. For a node $v \in V$ and topic $t \in T$, $I(v, t) = 1$ if and only if node v is subscribed to topic t , and $I(v, t) = 0$ otherwise.

For a set of nodes V , an overlay network $G(V, E)$ is an undirected graph on the node set V with edge set $E \subseteq V \times V$. For a topic $t \in T$, let $V_t = \{v \in V | I(v, t) = 1\}$. Given a topic $t \in T$ and an overlay network $G(V, E)$, the number of topic-connected components of G for topic t is equal to the number of connected components of the subgraph of G induced by V_t . An overlay network G is *topic-connected* if and only if it has one topic-connected component for each topic $t \in T$. The diameter of a graph is the length of the longest shortest path in the graph. The degree of a node v in an overlay network $G(V, E)$ is equal to the total number of edges adjacent to v in G .

Minimum Maximum Degree Topic-Connected Overlay (MinMax-TCO) Problem: Given a collection of nodes V , a set of topics T , and the node interest assignment I , connect the nodes in V into a topic-connected overlay network G which has least possible maximum degree.

4.2.2. MinAv-TCO problem and Greedy Merge(GM) Algorithm

The MinAv-TCO problem was introduced by Chockler et. al. [Chockler et al., 2007a] in which they aim at minimizing the average node degree. In this section we present a formal

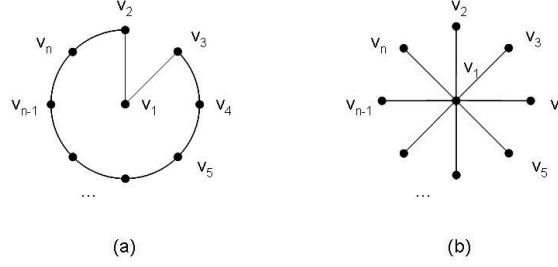


Figure 13. (a) Overlay with optimal max degree (b) Overlay constructed by GM

definition of the MinAv-TCO problem and outline the main techniques in the corresponding Greedy Merge (GM) algorithm, which will be useful for our approach to MinMax-TCO. We start with a formal definition of the MinAv-TCO problem.

Minimum Topic Connected Overlay Problem (MinAv-TCO): Given a collection of nodes V , a set of topics T , and a node interest assignment I , connect the nodes in V into a topic-connected overlay network G which has the least possible total number of edges (and hence the least possible average node degree).

The Greedy Merge (GM) Algorithm [Chockler et al., 2007a]: Initially we have the set of nodes V and no edges between the nodes. At each step, add the edge which maximally reduces the total number of topic-connected components.

The GM algorithm does not work well for the MinMax-TCO problem: The approximation ratio on the maximum degree obtained by the GM algorithm may be as bad as $\Theta(n)$, as we show in the lemma below.

Lemma 4.2.1 *The GM algorithm can only guarantee an approximation ratio of at least $\Theta(n)$ for the MinMax-TCO problem, where n is number of nodes in the pub/sub system.*

Proof. Consider the example where we have n nodes $v_1, v_2, \dots, v_n$, and n topics $T =$

$\{t_1, t_2, \dots, t_n\}$. Node v_1 is interested in all topics in T and each v_i is interested in $T - t_i$, $2 \leq i \leq n$. The GM algorithm would produce an overlay network with max degree $n - 1$. The overlay network in Figure 13 (b), where $E = \{(v_1, v_i) | 1 < i \leq n\}$, would result from the GM algorithm – the maximum degree of this overlay network is $n - 1$. The optimal solution for the MinMax-TCO on the same configuration for the nodes V is the overlay network $G(V, E')$, where $E' = \{(v_i, v_{i+1}) | 3 \leq i < n\} \cup \{(v_n, v_2), (v_1, v_2), (v_1, v_3)\}$ (see Figure 13 (a)), which has maximum degree 2. Hence the approximation ratio of the GM algorithm can be as large as $(n - 1)/2 = \Theta(n)$.

□

4.2.3. Overlay Design Algorithm

In this section we present our overlay design algorithm (MinMax-ODA) for the MinMax-TCO problem. MinMax-ODA starts with the overlay network $G(V, \emptyset)$. At each iteration of MinMax-ODA, a maximum weight edge — where the weight of an edge (u, v) is given by the reduction on the number of topic-connected components which would result from the addition of (u, v) to the current overlay network — among the ones which minimally increases maximum degree of the current graph is added to edge set of the overlay network. Let $NC(V, E)$ denote total number of topic connected components in the overlay network given by (V, E) .

Steps 1-6 of MinMax-ODA build an initial weighted graph $G'(V, E', w)$ on V , where $E' = V \times V$ and $w(\{u, v\})$ is equal to the amount of decrease in the number of topic-connected components resulting from the addition of the edge (u, v) to the current

Algorithm 11 Minimum Maximum Degree Overlay Design Algorithm (MinMax-ODA)

```

1: OverlayEdges  $\leftarrow \emptyset$ 
2:  $V \leftarrow$  Set of all nodes
3:  $G'(V, E') \leftarrow$  Complete graph on  $V$ 
4: for  $\{u, v\} \in E'$  do
5:    $w\{u, v\} \leftarrow$  Number of topics that both of nodes  $u$  and  $v$  have
6: end for
7: while  $G(V, \text{OverlayEdges})$  is not topic-connected do
8:   Find maximum-weighted edge  $e$  on  $G'(V, E', w)$  among the ones which increase the
   maximum degree of  $G(V, \text{OverlayEdges})$  minimally.
9:    $\text{OverlayEdges} = \text{OverlayEdges} \cup e$ 
10:   $E' \leftarrow E' - e$ 
11:  for  $\{u, v\} \in E'$  do
12:     $w\{u, v\} \leftarrow \text{NC}(V, \text{OverlayEdges}) - \text{NC}(V, \text{OverlayEdges} \cup \{u, v\})$ 
13:  end for
14: end while

```

overlay network (represented by the edges in OverlayEdges). Initially, this amount will be equal to the number of topics that nodes u and v have in common.

While at a first glance MinMax-ODA may look very similar to GM, it actually can be shown to work in phases, where in each phase a collection of edges that form matching of the nodes in the pub/sub system that connects close to the maximum number of connected components for the different topics is selected, as we explain below. Such a matching decomposition of the selected edges, crucial for our approximation ratio analysis, was not possible for GM.

At each iteration of the while loop, a maximum weight edge among the ones which increase the maximum degree of the current graph minimally is added to the set of overlay edges. Note that the addition of an edge to OverlayEdges can either increase the maximum degree by 1 or not increase it at all. For ease of explanation, assume that we have an even number of nodes in the pub/sub system. Since we start with all nodes having equal degree

(equal to 0), MinMax-ODA will first select a set of edges that increases the degree of every node to 1 (i.e., a matching), and then a set of edges (another matching) that increases the degree of each node to 2, etc. Note that some of these edges may have weight 0 (i.e., they do not really contribute to the construction of the topic-connected components), but they will not affect the final solution obtained (the 0-weight edges will be discarded at the end of the algorithm). The crux in the analysis of this algorithm is to show that each of these matchings will reduce the number of connected components by a “large” amount.

Before we proceed in proving the approximation ratio on the maximum degree guaranteed by MaxMin-ODA, we prove that the algorithm terminates in $O(|V|^4|T|)$ time.

Lemma 4.2.2 *The MinMax-ODA algorithm terminates within $O(|V|^2)$ iterations on the while loop.*

Proof. At each iteration of the while loop, at least one edge is added to the current overlay network. Hence the algorithm will terminate in at most $O(|V|^2)$ iterations.

□

Lemma 4.2.3 *The running time of MinMax-ODA is $O(|V|^4|T|)$.*

Proof. The weight initialization takes $O(|V|^2|T|)$ time. Updating the weight of each of the remaining edges takes $O(1)$ time ([Chockler et al., 2007a], Lemma 6.4). Finding the edge with max weight will take at most $O(|V|^2)$ time. Since total weight of the edges is $O(|V|^2|T|)$ at the beginning and greater than 0 at the end, MinMax-ODA takes $O(|V|^2|T|) * O(|V|^2) = O(|V|^4|T|)$ time.

□

4.2.4. Approximation Ratio

In this section, we will prove that our overlay design algorithm (MinMax-ODA) approximates the MinMax-TCO problem within a logarithmic factor.

Theorem 4.2.4 *The overlay network output by MinMax-ODA has maximum node degree within a factor of $O(\log(\sum_{v \in V} |\{t \in T \mid I(v, t) = 1\}|))$ from the minimum possible maximum node degree for any topic-connected overlay network on V .*

Proof. At a high level, the proof follows the general lines as the proof of the logarithmic approximation ratio for the classic set cover problem (which was also the basis for the approximation ratio proof of the GM algorithm for the MinAv-TCO problem [Chockler et al., 2007a]). However, before we can apply the set cover framework, we first need to carefully show that MinMax-ODA works as if we had many applications of a greedy matching algorithm that aims at reducing the number of connected components maximally and then relate our network overlay construction to a matching decomposition of an optimal (i.e., a minimum maximum degree) overlay network.

Assume we have an instance of the MinMax-TCO problem and that $G(V, E_{opt})$ is an optimum solution for this instance with maximum degree d_{opt} . We will use the following well-known result in graph theory for the proof.

Lemma 4.2.5 *Given a graph $G(V, E)$ with maximum degree d , we can divide the edge set E into $k = d + 1$ matchings M_i , $1 \leq i \leq k$.*

Proof. We can color the edges of any graph with $d + 1$ colors such that any adjacent edge will have different color. This is Vizing's Edge Coloring Theorem (Theorem 5.3.2 in

[Diestel, 2000]). Since each coloring class is a matching, we can divide the edge set into $d + 1$ matchings.

□

Using the lemma above, we can divide the edge set E_{opt} of the optimum solution into $k = d_{opt} + 1$ matchings $M_i, 1 \leq i \leq k$.

At the beginning of the algorithm, the total number of connected components is $C_{start} = \sum_{v \in V} |\{t \in T | I(v, t) = 1\}|$ and at the end $C_{end} = |\{t | t \in T \text{ and } \exists v \in V \text{ such that } I(v, t) = 1\}|$. Note that since we count the connected components for each topic separately, once we get down to C_{end} components, there must exist *exactly one* component for each active topic t (i.e., each t such that there exists some v with $I(v, t) = 1$) — i.e., the overlay network is topic-connected.

For ease of explanation, assume that we have an even number of nodes in the pub/sub system (if there are an odd number of nodes, one can always add a “dummy” node which is not subscribed to any topic, without affecting the final solution; or we can handle small deviations from a perfect matching decomposition in the analysis). At each iteration of the while loop, a maximum weight edge among the ones which increases the maximum degree of the current graph minimally is added to the set of overlay edges. At start all nodes have degree 0. After a number of iterations the edges added will form a perfect matching and then the next edge added will increase the max degree of the graph by 1.

Let S_i be the edge set of the i^{th} matching added to the set by the algorithm MinMax-ODA, $1 \leq i \leq k$. Let n_i be total number of connected components before we add i^{th}

matching, so $n_1 = C_{start}$. Let $SA_i = S_1 \cup S_2 \cup \dots \cup S_{i-1}$ be the union of all matchings found before the algorithm starts adding the i -th matching.

The following lemma proves that each matching S_i chosen by our algorithm decrease the current total number of connected components at least $(1/3)$ - optimally.

Lemma 4.2.6 *The matching S_i reduces the total number of connected components of $G(V, SA_i)$ by at least $1/3$ of any optimal matching which reduces by maximum amount.*

Proof. Let P be the edge set of the matching which reduces the total number of connected components of the $G(V, SA_i)$ by the maximum amount, which we denote by c . Let $Q = \{e_1, e_2, \dots, e_j\}$ be the edge set of the matching S_i that our algorithm finds. Let $e_l = u_l v_l$ for $1 \leq l \leq j$. For e_a and e_b , if $a < b$, then e_a is found before e_b by our algorithm. Let Q reduce the total number of connected components of the $G(V, SA_i)$ by c' . Let $G_0 = G(V, SA_i)$ and $G_l = G_{l-1} \cup e_l$, for $1 \leq l \leq j$. Let e_l reduce the total number of connected components of G_{l-1} by y_l . Then,

$$c' = \sum_{1 \leq l \leq j} y_l \quad (1)$$

$$y_a \geq y_b, \text{ for } 1 \leq a \leq b \leq j \quad (2)$$

Let X_l be the set of edges in P which are incident to u_l or v_l , $1 \leq l \leq j$, but not incident to $u_{l'}$ or $v_{l'}$ $1 \leq l' \leq l-1$. Thus, X_l will have zero or one or two edges for $1 \leq l \leq j$. Let $P_0 = P$ and $P_l = P_{l-1} - X_l$ for $1 \leq l \leq j$. Since Q is a maximal matching, $P_j = \emptyset$. Let X_l reduce the total number of connected components of G_{l-1} by

x_l for $1 \leq l \leq j$. Let P_l reduce the total number of connected components of G_l by c_l for $0 \leq l \leq j$.

If X_l has two edges, then our algorithm did not choose one of these two edges at that step and choose e_l instead, $0 \leq l \leq j$. Since our algorithm greedily choose the edges, e_l reduces the total number of connected components of G_{l-1} by at least as much as each of the edges in X_l . Hence, $y_l \geq x_l/2$. Similarly, if X_l has one or zero edges, then $y_l \geq x_l$. So,

$$y_l \geq \frac{x_l}{2}, 1 \leq l \leq j$$

$$\Rightarrow \sum_{1 \leq l \leq j} y_l \geq \frac{1}{2} \sum_{1 \leq l \leq j} x_l \quad (3)$$

Since $P_{l+1} = P_l - X_{l+1}$ and $G_{l+1} = G_l \cup e_{l+1}$, $0 \leq l \leq j-1$, the amount that P_l reduces the total number of connected components of G_l is smaller than sum of the amount that P_{l+1} reduces the total number of connected components of G_{l+1} and the amount that e_{l+1} reduces the total number of connected components of G_l and the amount X_{l+1} reduces the total number of connected components of G_l . Hence,

$$c_0 = c, c_j = 0 \quad (4)$$

$$c_{l+1} \geq c_l - (x_{l+1} + y_{l+1}) \text{ for } 0 \leq l \leq j-1 \quad (5)$$

If we add all the inequalities (4) and (5), we will have

$$\sum_{1 \leq l \leq j} x_l + \sum_{1 \leq l \leq j} y_l \geq c \quad (6)$$

From the inequalities (3) and (6), we will have

$$3 \sum_{1 \leq l \leq j} y_l \geq c \quad (7)$$

From the inequalities (1) and (7), we will have

$$c' \geq c/3$$

□

Before MinMax-ODA starts adding the i^{th} matching, we have n_i components and we know that if we add all the $k = d_{opt} + 1$ matchings $M_j - SA_i$, $1 \leq j \leq k$, to the current solution, the total number of connected components will be reduced to C_{end} . Therefore, there exists a matching $M_j - SA_i$ which decreases the total number of connected components by at least $(n_i - C_{end})/k$. Since our algorithm always finds at least a $(1/3)$ -optimal matching (Lemma 4.3.7), the matching S_i that our algorithm uses must decrease the total number of connected components at that time by at least $(1/3)$ of this amount. Therefore,

$$\begin{aligned}
n_i - n_{i+1} &\geq (n_i - C_{end})/(3k) \\
\Rightarrow n_{i+1} - C_{end} &\leq (1 - 1/(3k))(n_i - C_{end}).
\end{aligned}$$

Hence, the number of iterations for our algorithm MinMax-ODA is less than or equal to the smallest m which satisfies

$$\begin{aligned}
1 &> (n_1 - C_{end})(1 - 1/(3k))^m \\
\Rightarrow m &\leq 3k \ln(C_{start} - C_{end}) \\
\Rightarrow m &\leq 3k \ln(C_{start}) \\
\Rightarrow m &\leq 3(d_{opt} + 1) * \ln(C_{start})
\end{aligned}$$

□

4.2.5. Hardness of MinMax-TCO Problem

In this section, we will show that no polynomial time algorithm can approximate the MinMax-TCO problem within a constant factor.

Theorem 4.2.7 *There exists no polynomial time algorithm that approximates the MinMax-TCO problem within a constant factor unless $P=NP$.*

Proof. The proof follows the proof of Theorem 5.3 in [Chockler et al., 2007a]. In fact, a more careful observation of the proof of Theorem 5.3 in [Chockler et al., 2007a] shows that the proof basically shows the hardness of approximation of the maximum degree (in [Chockler et al., 2007a], the authors consider the problem of approximating the

average degree). For the sake of completeness, we present here the sketch of the proof of Theorem 5.3 [Chockler et al., 2007a] when directly applied to maximum degree. We first define the single node version of the MinMax-TCO problem.

Single node version of the MinMax-TCO problem (SN-MinMax-TCO): Given V, T, I , a node $v \in V$, connect the nodes in V into an overlay network G which has least possible degree for node v and G is topic-connected.

We can prove that SN-MinMax-TCO is NP-hard by reducing the minimum set cover problem to this problem. Then we show how to reduce the SN-MinMax-TCO problem to the MinMax-TCO problem, thus showing that the MinMax-TCO problem is NP-hard.

Now, we are ready to prove our inapproximability result. We will use the same reduction as in the proof of Theorem 5.3 in [Chockler et al., 2007a]. Assume that there is an algorithm A which approximates MinMax-TCO problem within a constant factor. We will show we can use this algorithm A to find a constant factor approximation to the minimum set cover problem, which is known to be impossible unless $P = NP$.

Given an instance of $SN - MinMax - TCO(V, T, I, v)$ problem, construct the corresponding instance of $MinMax - TCO(V', T', I')$ problem as in the construction of proof of Lemma 5.1 [Chockler et al., 2007a]. Let d_{opt} denote the maximum degree of an optimal solution for this instance of $SN - MinMax - TCO(V, T, I, v)$ problem. As shown in proof of Lemma 5.1 [Chockler et al., 2007a], the corresponding $MinMax - TCO(V', T', I')$ problem also has a solution with degree d_{opt} . So, the algorithm A will find a solution to the

corresponding $MinMax - TCO(V', T', I')$ problem with degree at most $c * d_{opt}$. With using this solution, we can construct a solution to $SN - MinMax - TCO(V, T, I, v)$ problem with degree at most $c * d_{opt}$ as in Lemma 5.1 [Chockler et al., 2007a]. Thus, we have a constant approximation algorithm B for $SN - MinMax - TCO(V, T, I, v)$ problem.

Given an instance of the minimum set cover problem (U, S) , construct the corresponding $SN - MinMax - TCO(V, T, I, v)$ problem same as in the construction of proof of Lemma 5.2 [Chockler et al., 2007a]. Denote s_{opt} an optimal solution for this instance of minimum set cover problem. As shown in proof of Lemma 5.2 [Chockler et al., 2007a], the corresponding $SN - MinMax - TCO(V, T, I, v)$ problem has a solution with degree s_{opt} . So, the algorithm B will find a solution to the corresponding $SN - MinMax - TCO(V, T, I, v)$ problem with degree at most $c' * s_{opt}$. With using this solution, we can construct a solution to minimum set cover problem with number of sets at most $c' * s_{opt}$ as in Lemma 5.2 [Chockler et al., 2007a]. Thus, we have a constant approximation algorithm C for the minimum set cover problem.

□

Since it is trivial to show that MinMax-TCO is in NP, it follows

Corollary 4.2.8 *The MinMax-TCO problem is NP-complete.*

4.2.6. Experimental Results

The GM algorithm [Chockler et al., 2007b] and our MinMax-ODA algorithm are implemented in Java. These two algorithms are compared according to maximum degree and

average degree in the resulting overlay graphs. Experimental results show that MinMax-ODA improves the maximum degree of the overlay network drastically at the cost of a small increase on the average degree.

4.2.6.1. MAXIMUM NODE DEGREE

For these experiments, the number of nodes varies between 100 to 1000. In the first experiment (Figure 14(a)), the number of topics is 100 and in the second experiment (Figure 14(b)) the number of topics is 200. We fixed number of subscriptions to $s = 10$. Each node is interested in each topic uniformly at random. This experimental setting is similar to previous studies [Chockler et al., 2007b, Ramasubramanian et al., 2006].

Figure 14(a) is a comparison of GM and MinMax-ODA algorithm according to the maximum degree. The maximum degree of the graph decreases for MinMax-ODA algorithm when the number of nodes increases since MinMax-ODA algorithm can find edges with higher correlation as the number of nodes increases. Interestingly, the maximum degree of the graph increases for the GM algorithm as the number of nodes increases. Basically as the number of nodes increase, the GM algorithm will assign more edges to same the nodes since now we have more nodes with higher correlation for each node. When we compare the results of GM and MinMax-ODA algorithm, MinMax-ODA improves GM by factor 3 on average (Figure 14(a)).

The same results are valid for Figure 14(b). When we compare Figure 14(a) and Figure 14(b), max node degree increases slightly for both GM and MinMax-ODA since edges will have less correlation when we increase the number of topics.

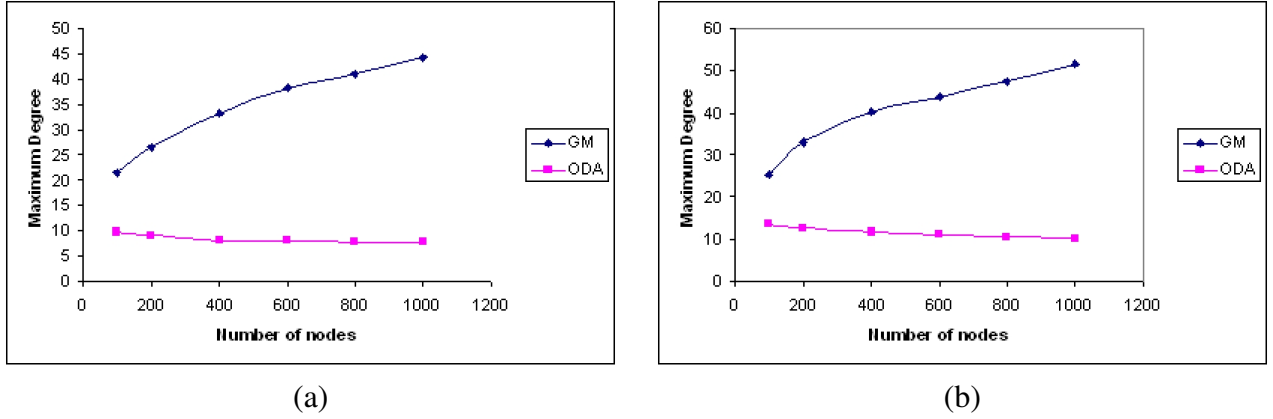


Figure 14. Maximum node degree for GM and MinMax-ODA (a) Number of topics is 100 (b) Number of topics is 200

4.2.6.2. AVERAGE NODE DEGREE

Experimental setting is same as previous subsection. Figure 15(a) is comparison of GM and MinMax-ODA algorithm according to average degree. The average degree of the graph decreases for both GM and MinMax-ODA algorithms when the number of nodes increases since algorithms can find edges with higher correlation when the number of nodes increases. When we compare the results of GM and MinMax-ODA algorithm, GM is slightly better than MinMax-ODA, 7% on average, (Figure 15(a)). Similar results are valid for Figure 15(b).

4.2.6.3. SUBSCRIPTION SIZE

In these experiment, the number of nodes and the number of topics are fixed to 100. The subscription size varies between 10 to 50. Each node is interested in each topic uniformly randomly. Figure 16(a) is the comparison of GM and MinMax-ODA algorithm according to the maximum degree. The maximum degree of the overlay network decreases for

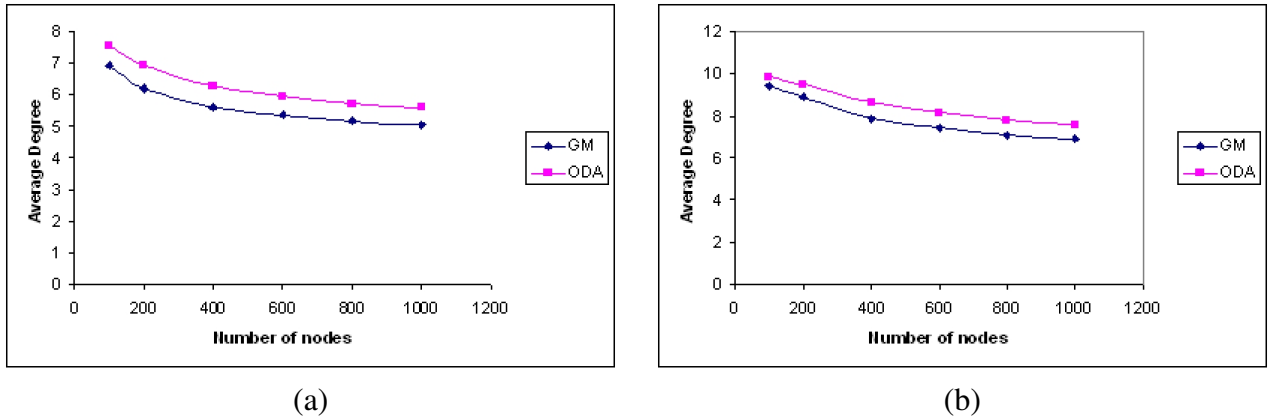


Figure 15. Average node degree for GM and MinMax-ODA (a) Number of topics is 100 (b) Number of topics is 200

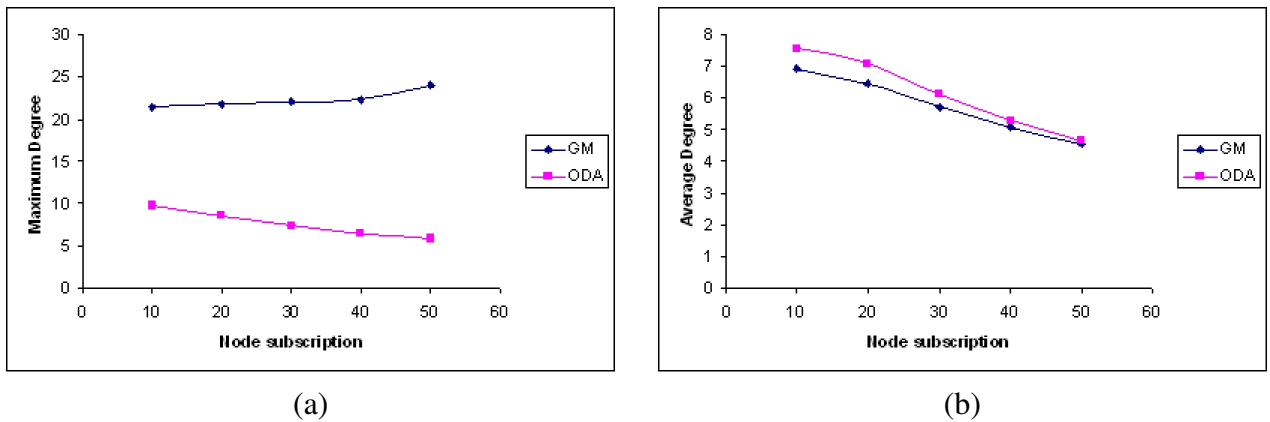


Figure 16. Average and maximum node degree for different subscription size (Number of nodes and number of topics is 100)

the MinMax-ODA algorithm as the subscription size increases since the MinMax-ODA algorithm can find edges with higher correlation when the subscription size increases. Interestingly, the maximum degree of the overlay network increases for GM algorithm as the subscription size increases. When the subscription size increases, GM algorithm will assign more edges to the same nodes since now we have more nodes with higher correlation for each node. When we compare the results of GM and MinMax-ODA algorithms,

MinMax-ODA improves GM by factor a 4 on average (Figure 14(a)).

Figure 16(b) is comparison of GM and MinMax-ODA algorithm according to the average degree. The average degree of the overlay network decreases for both GM and MinMax-ODA algorithms when subscription size increases since algorithms can find edges with more correlation. When we compare the results of GM and MinMax-ODA algorithm, GM is slightly better than MinMax-ODA, %10 on average, (Figure 16(b)).

4.3. Parameterized Maximum and Average Degree Approximation In Topic-Based Publish-Subscribe Overlay Network Design

This section is organized as follows. In Section 4.3.1, we prove MinMax-ODA is $\Theta(n)$ approximation for the Low-TCO problem. Section 4.3.2 presents our topic-connected overlay design algorithm Low-ODA, whose approximation ratio is proved in Section 4.3.3.

4.3.1. Minimizing The Maximum Degree Only

The MinMax-TCO problem and corresponding MinMax-ODA algorithm aim at minimizing the maximum node degree. Unfortunately, while the MinMax-ODA algorithm produces a logarithmic approximation on the maximum node degree, it fails to approximate well the average node degree of a topic-connected overlay network: The approximation ratio on the average degree obtained by the MinMax-ODA algorithm may be as bad as $\Theta(n)$, as we show in the lemma below.

Lemma 4.3.1 *The MinMax-ODA algorithm can only guarantee an approximation ratio of $\Theta(n)$ on the average node degree, where n is number of nodes in the pub/sub system.*

Proof. Consider the example where we have n nodes $v_1, v_2, \dots, v_n$, and n^2 topics $T = \{t_{i,j} | 1 \leq i, j \leq n\}$. Node v_1 is interested in all topics in T and each v_i is interested in $t_{i,j}$ and $t_{j,i}$, $2 \leq i \leq n$, $1 \leq j \leq n$. W.l.o.g., assume that n is even. The MinMax-ODA algorithm will produce an overlay network with $n(n-2)/2$ edges, by repeatedly connecting a maximal matching of the nodes $v_1, \dots, v_n$, $n-2$ times. The optimal overlay network with minimum number of edges is $E = \{(v_1, v_i) | 1 < i \leq n\}$, – the number of edges of this overlay network is $n-1$. Hence the approximation ratio of the MinMax-ODA algorithm can be as large as $(n(n-2)/2)/(n-1) = \Theta(n)$.

□

4.3.2. Low Degree Overlay Design Algorithm (Low-ODA)

In this section we present our overlay design algorithm (Low-ODA) for the Low-TCO problem. The weight of an edge (u, v) is given by the reduction on the number of topic-connected components which would result from the addition of (u, v) to the current overlay network. Let $1 \leq k \leq n$. Low-ODA starts with the overlay network $G(V, \emptyset)$. At each iteration of Low-ODA, the algorithm considers two edges:

e_1 : a maximum weight edge among the ones which minimally increases maximum degree of the current graph

e_2 : a maximum weight edge

If the weight of edge e_1 is greater than the weight of e_2 divided by k , edge e_1 is added to edge set of the overlay network; otherwise edge e_2 is added.

Let $NC(V, E)$ denote total number of topic connected components in the overlay network given by (V, E) .

Algorithm 12 Low Degree Overlay Design Algorithm (Low-ODA)

```

1: OverlayEdges  $\leftarrow \emptyset$ 
2:  $V \leftarrow$  Set of all nodes
3:  $G'(V, E') \leftarrow$  Complete graph on  $V$ 
4: for  $\{u, v\} \in E'$  do
5:    $w\{u, v\} \leftarrow$  Number of topics that both of nodes  $u$  and  $v$  have
6: end for
7: while  $G(V, \text{OverlayEdges})$  is not topic-connected do
8:   Let  $e_1$  be a maximum weight edge in  $G'(V, E', w)$  among the ones which increase
     the maximum degree of  $G(V, \text{OverlayEdges})$  minimally.
9:   Let  $e_2$  be a maximum weight edge in  $G'(V, E', w)$ 
10:  if  $w(e_1) \geq w(e_2)/k$  then
11:     $e = e_1$ 
12:  else
13:     $e = e_2$ 
14:  end if
15:   $\text{OverlayEdges} = \text{OverlayEdges} \cup e$ 
16:   $E' \leftarrow E' - e$ 
17:  for  $\{u, v\} \in E'$  do
18:     $w\{u, v\} \leftarrow NC(V, \text{OverlayEdges}) - NC(V, \text{OverlayEdges} \cup \{u, v\})$ 
19:  end for
20: end while

```

Steps 1-6 of Low-ODA build an initial weighted graph $G'(V, E', w)$ on V , where $E' = V \times V$ and $w(\{u, v\})$ is equal to the amount of decrease in the number of topic-connected components resulting from the addition of the edge (u, v) to the current overlay network (represented by the edges in OverlayEdges). Initially, this amount will be equal to the number of topics that nodes u and v have in common.

At each iteration of the while loop, two edges are considered: an edge (e_1) with

maximum weight among the edges in E' that increase the maximum degree of the current graph minimally and an edge (e_2) with maximum weight in all of E' . If weight of the first one (e_1) is greater than or equal to weight of the second one (e_2) over k , e_1 is added to the set of overlay edges; otherwise e_2 is added. Note that the addition of an edge to `OverlayEdges` can either increase the maximum degree by 1 or not increase it at all. The crux in the analysis of this algorithm is to show that each of the edges will reduce the number of connected components by a “large” amount without increasing the maximum degree by too much.

While at a first glance the Low-ODA algorithm may seem like a trivial combination of the GM and MinMax-ODA algorithms, the analysis show that such a combination is far from trivial: Once we allow the algorithm (Low-ODA) to select some edges based solely on their weight (edges of type e_2), the “perfect matching” behavior of the edges selected by MinMax-ODA (basically one could show that the first $n/2$ edges selected by MinMax-ODA formed a perfect matching in G' , as did the second set of $n/2$ edges, etc.) is no longer valid and the approximation ratio analysis used in MinMax-ODA (which heavily relied on this perfect matching decomposition of the edges selected) can no longer be directly used here.

Before we proceed in proving the approximation ratio on the maximum degree and the approximation ratio on the average degree guaranteed by Low-ODA, we prove that the algorithm terminates in $O(|V|^4|T|)$ time.

Lemma 4.3.2 *The Low-ODA algorithm terminates within $O(|V|^2)$ iterations on the while loop.*

Proof. At each iteration of the while loop, at least one edge is added to the current overlay network. Hence the algorithm will terminate in at most $O(|V|^2)$ iterations.

□

Lemma 4.3.3 *The running time of Low-ODA is $O(|V|^4|T|)$.*

Proof. The weight initialization takes $O(|V|^2|T|)$ time. Updating the weight of each of the remaining edges takes $O(1)$ time ([Chockler et al., 2007a], Lemma 6.4). Finding the edge with max weight will take at most $O(|V|^2)$ time. Since total weight of the edges is $O(|V|^2|T|)$ at the beginning and greater than 0 at the end, Low-ODA takes $O(|V|^2|T|) * O(|V|^2) = O(|V|^4|T|)$ time.

□

4.3.3. Approximation Ratio

In this section, we will prove that our overlay design algorithm (Low-ODA) approximates the average degree by factor $O(k * \log(\sum_{v \in V} |\{t \in T | I(v, t) = 1\}|))$ and the maximum degree by factor $O((n/k) * \log(\sum_{v \in V} |\{t \in T | I(v, t) = 1\}|))$.

As we mentioned in the previous section, the main challenge in the analysis is to overcome the fact that we can no longer think of the algorithm as selecting a sequence of perfect matchings of the nodes in V when bounding the approximation ratio on the maximum degree (the analysis of MinMax-ODA algorithm heavily relied on this “perfect matching” behavior).

Theorem 4.3.4 *The overlay network output by Low-ODA has average node degree within a factor of $O(k * \log(\sum_{v \in V} |\{t \in T | I(v, t) = 1\}|))$ from the minimum possible average node degree for any topic-connected overlay network on V .*

Proof. The proof follows the general lines as the proof of the logarithmic approximation ratio for the classic set cover problem (which was also the basis for the approximation ratio proof of the GM algorithm for the MinAv-TCO problem [Chockler et al., 2007a]). Assume we have an instance of the Low-TCO problem and that $G(V, E_{opt})$ is a solution for this instance with minimum number of edges. Let $|E_{opt}| = m$. At the beginning of the algorithm, the total number of connected components is $C_{start} = \sum_{v \in V} |\{t \in T | I(v, t) = 1\}|$ and at the end $C_{end} = |\{t | t \in T \text{ and } \exists v \in V \text{ such that } I(v, t) = 1\}|$. Note that since we count the connected components for each topic separately, once we get down to C_{end} components, there must exist *exactly one* component for each active topic t (i.e., each t such that there exists some v with $I(v, t) = 1$) — i.e., the overlay network is topic-connected.

Let e_i be the i^{th} edge added to the set by the algorithm Low-ODA. Let n_i be total number of connected components before we add i^{th} edge, so $n_1 = C_{start}$. Let $S_i = \{e_1, e_2, \dots, e_{i-1}\}$ be the set of all edges found before the algorithm starts adding the i -th edge. Before Low-ODA starts adding the i^{th} edge, we have n_i components and we know that if we add all the edges $E_{opt} - S_i$, to the current solution, the total number of connected components will be reduced to C_{end} . Since $|E_{opt} - S_i| \leq m$, there exists an edge which decreases the total number of connected components by at least $(n_i - C_{end})/m$. Since our algorithm always adds at least a $(1/k)$ -optimal edge, the edge e_i that our algorithm adds must decrease the total number of connected components at that time by at least $(1/k)$ of

this amount. Therefore,

$$n_i - n_{i+1} \geq (n_i - C_{end})/(m * k) \Rightarrow n_{i+1} - C_{end} \leq (1 - 1/(m * k))(n_i - C_{end}).$$

Hence, the number of iterations for our algorithm Low-ODA is less than or equal to the smallest z which satisfies

$$\begin{aligned} 1 &> (n_1 - C_{end})(1 - 1/(m * k))^z \Rightarrow z \leq m * k \ln(C_{start} - C_{end}) \\ \Rightarrow z &\leq m * k \ln(C_{start}). \end{aligned}$$

□

Theorem 4.3.5 *The overlay network output by Low-ODA has maximum node degree within a factor of $O((n/k) * \log(\sum_{v \in V} |\{t \in T | I(v, t) = 1\}|))$ from the minimum possible maximum node degree for any topic-connected overlay network on V .*

Proof. At a high level, the proof follows the general lines as the proof of the logarithmic approximation ratio for the classic set cover problem (which was also the basis for the approximation ratio proof of the MinMax-ODA algorithm for the MinMax-TCO problem [Onus and Richa, 2009b]).

However, before we can apply the set cover framework, we first need to carefully relate the sequence of edges selected by the Low-ODA algorithm to a sequence of optimal matchings which reduce the current number of connected components maximally. Note that we no longer can break the sequence of edges selected by our algorithm into a sequence of perfect matchings, as in the MinMax-ODA algorithm.

Assume we have an instance of the Low-TCO problem and that $G(V, E_{opt})$ is a

solution with minimum possible maximum degree for this instance. Let this maximum degree is d_{opt} . We will use the following well-known result in graph theory for the proof.

Lemma 4.3.6 ((Lemma 4 in [Onus and Richa, 2009b])) *Given a graph $G(V, E)$ with maximum degree d , we can divide the edge set E into $d + 1$ matchings M_i , $1 \leq i \leq (d + 1)$.*

Using the lemma above, we can divide the edge set E_{opt} of the optimum solution into $d_{opt} + 1$ matchings M_i , $1 \leq i \leq (d_{opt} + 1)$.

At the beginning of the algorithm, the total number of connected components is $C_{start} = \sum_{v \in V} |\{t \in T | I(v, t) = 1\}|$ and at the end $C_{end} = |\{t | t \in T \text{ and } \exists v \in V \text{ such that } I(v, t) = 1\}|$. Note that since we count the connected components for each topic separately, once we get down to C_{end} components, there must exist *exactly one* component for each active topic t (i.e., each t such that there exists some v with $I(v, t) = 1$) — i.e., the overlay network is topic-connected.

At start all nodes have degree zero. At each iteration of the while loop, a maximum weight edge among the ones which increases the maximum degree of the current graph minimally or a maximum weight edge is added to the set of overlay edges.

After a number of iterations, the weight of a maximum weight edge among the ones which increases the maximum degree of the current graph minimally will be less than the weight of a maximum weight edge over k and we will add this maximum weight edge to the graph and this edge will increase the maximum degree of the graph by 1.

Let S_i be the edge set containing all edges added by Low-ODA from the time an edge e' increased the maximum degree of G' from $i - 1$ to i until the last time an edge

is added to G' without increasing its maximum degree further (i.e., without increasing the maximum degree to $i + 1$).

Let $h = n/(2k) + 1$. Let $R_i = S_{h*(i-1)+1} \cup S_{h*(i-1)+2} \cup \dots \cup S_{h*(i-1)+h}$. Let $RA_i = R_1 \cup R_2 \cup \dots \cup R_{i-1}$ be the union of all edges added before the algorithm starts adding the set R_i . Let n_i be total number of connected components before the algorithm adds R_i , so $n_1 = C_{start}$.

The following lemma proves that each set R_i chosen by our algorithm decreases the current total number of connected components by at least $1/3$ of the number of current components connected by any optimal matching, where an optimal matching is one that reduces the current number of connected components maximally among all possible maximal matchings in G' . Note that a matching increases the maximum degree of G' by at most 1.

Lemma 4.3.7 *The set R_i reduces the total number of connected components of $G(V, RA_i)$ by at least $1/3$ of the number of current components connected by any optimal matching.*

Proof. Let P be the edge set of the matching which reduces the total number of connected components of the $G(V, RA_i)$ by the maximum amount, which we denote by c . Let $Q = \{e_1, e_2, \dots, e_j\}$ be the edge set of R_i . Let $e_l = u_l v_l$ for $1 \leq l \leq j$. For e_a and e_b , if $a < b$, then e_a is found before e_b by our algorithm. Let Q reduce the total number of connected components of the $G(V, RA_i)$ by c' . Let $G_0 = G(V, RA_i)$ and $G_l = G_{l-1} \cup e_l$, for $1 \leq l \leq j$. Let e_l reduce the total number of connected components of G_{l-1} by y_l . Then,

$$c' = \sum_{1 \leq l \leq j} y_l \quad (1)$$

Consider the case where e_l does not increase the maximum degree of current graph or $l = 1$ or e_l increases the maximum degree of current graph and there is no possible edge which does not increase maximum degree of the current graph, $1 \leq l \leq j$. Let X_l be the set of edges in P which are incident to u_l or v_l , $1 \leq l \leq j$, and not in $X_{l'}$, $1 \leq l' \leq l - 1$. Thus, X_l will have zero or one or two edges for $1 \leq l \leq j$.

Now consider the case where e_l increases the maximum degree of current graph and there are some edges which does not increase maximum degree of the current graph, $2 \leq l \leq j$. Let X_l be the set of the first k maximum weight edges in P which are not in $X_{l'}$, $1 \leq l' \leq l - 1$. If there are less than k elements in P which are not in $X_{l'}$, $1 \leq l' \leq l - 1$, X_l will only have these edges. If there are edges which are incident to u_l or v_l and not in $X_{l'}$, $1 \leq l' \leq l$, then replace any edges from X_l with those edges (note that there may be at most two edges of this kind).

Let $P_0 = P$ and $P_l = P_{l-1} - X_l$ for $1 \leq l \leq j$. Let X_l reduce the total number of connected components of G_{l-1} by x_l for $1 \leq l \leq j$. Let P_l reduce the total number of connected components of G_l by c_l for $0 \leq l \leq j$.

If there is an edge e_l which increases the maximum degree of current graph and there is no possible edge which does not increase maximum degree of the current graph, $2 \leq l \leq j$, then for each vertex of the graph, there is at least one edge $e_{l'}$ incident to this vertex, $1 \leq l' \leq l - 1$. So, union of sets $X_{l'}$, $1 \leq l' \leq l - 1$, contains all the edges in P . Thus, $P_j = \emptyset$. Now, consider the case when there is no edge which satisfies these

properties (so, when algorithm chooses an edge e_l which increases the maximum degree, there is always an edge which does not increase maximum degree of the current graph). Since R_i contains h sets of $S_{i'}$, there are $h - 1 = n/(2k)$ edges which increases maximum degree of the current graph. So $(n/2k)$ of X_l sets has k edges in it (if X_l has less than k edge than all edges of set P are already in one of sets $X_{l'}$, $1 \leq l' \leq l - 1$, so $P_j = \emptyset$). Union of sets X_l has at least $(n/2k) * k = n/2$ edges. Since P is a matching, it has at most $n/2$ edges. So, $P_j = \emptyset$. Hence,

$$c_0 = c, c_j = 0 \quad (2)$$

Consider the case where e_l does not increase the maximum degree of current graph or $l = 1$ or e_l increases the maximum degree of current graph and there is no possible edge which does not increase maximum degree of the current graph, $1 \leq l \leq j$. If X_l has two edges, then our algorithm did not choose one of these two edges at that step and choose e_l instead, $0 \leq l \leq j$. Since our algorithm greedily choose the edges, e_l reduces the total number of connected components of G_{l-1} by at least as much as each of the edges in X_l . Hence, $y_l \geq x_l/2$. Similarly, if X_l has one or zero edges, then $y_l \geq x_l$.

Now consider the case where e_l increases the maximum degree of current graph and there are some edges which does not increase maximum degree of the current graph, $2 \leq l \leq j$. X_l has at most k edges. Our algorithm did not choose one of these k edges at that step and choose e_l instead, $0 \leq l \leq j$. Since our algorithm greedily choose the edges, e_l reduces the total number of connected components of G_{l-1} by at least as much as k

times any of the edges in X_l . Since X_l has at most k edges, $y_l \geq x_l$. So,

$$y_l \geq \frac{x_l}{2}, 1 \leq l \leq j \quad \Rightarrow \quad \sum_{1 \leq l \leq j} y_l \geq \frac{1}{2} \sum_{1 \leq l \leq j} x_l \quad (3)$$

Since $P_{l+1} = P_l - X_{l+1}$ and $G_{l+1} = G_l \cup e_{l+1}$, $0 \leq l \leq j-1$, the amount that P_l reduces the total number of connected components of G_l is smaller than sum of the amount that P_{l+1} reduces the total number of connected components of G_{l+1} and the amount that e_{l+1} reduces the total number of connected components of G_l and the amount X_{l+1} reduces the total number of connected components of G_l . Hence,

$$c_{l+1} \geq c_l - (x_{l+1} + y_{l+1}) \text{ for } 0 \leq l \leq j-1 \quad (4)$$

If we add all the inequalities (2) and (4), we will have

$$\sum_{1 \leq l \leq j} x_l + \sum_{1 \leq l \leq j} y_l \geq c \quad (5)$$

From the inequalities (3) and (5), we will have

$$3 \sum_{1 \leq l \leq j} y_l \geq c \quad (6)$$

From the inequalities (1) and (6), we will have

$$c' \geq c/3$$

□

Before Low-ODA starts adding the set R_i , we have n_i components and we know that if we add all the $(d_{opt} + 1)$ matchings $M_j - RA_i$, $1 \leq j \leq (d_{opt} + 1)$, to the current solution, the total number of connected components will be reduced to C_{end} . Therefore, there exists a matching $M_j - RA_i$ which decreases the total number of connected components by at least $(n_i - C_{end})/(d_{opt} + 1)$. Since our algorithm always finds the set R_i which reduces the total number of connected components of $G(V, RA_i)$ by at least $1/3$ of any optimal matching which reduces by maximum amount (Lemma 4.3.7), the set R_i that our algorithm uses must decrease the total number of connected components at that time by at least $(1/3)$ of this amount. Therefore,

$$\begin{aligned} n_i - n_{i+1} &\geq (n_i - C_{end})/(3(d_{opt} + 1)) \\ \Rightarrow \quad n_{i+1} - C_{end} &\leq (1 - 1/(3(d_{opt} + 1)))(n_i - C_{end}). \end{aligned}$$

Hence, the number of iterations for our algorithm Low-ODA is less than or equal to the smallest m which satisfies

$$\begin{aligned} 1 &> (n_1 - C_{end})(1 - 1/(3(d_{opt} + 1)))^m \Rightarrow m \leq 3(d_{opt} + 1)\ln(C_{start} - C_{end}) \\ \Rightarrow \quad m &\leq 3(d_{opt} + 1)\ln(C_{start}) \end{aligned}$$

Since $h = n/(2k) + 1$, the maximum degree of resulting graph is less than or equal to

$$(n/(2k) + 1) * 3(d_{opt} + 1) * \ln(C_{start}).$$

□

4.4. Constructing Constant Diameter Overlays for Publish-Subscribe

In this section, we study a new optimization problem that constructs a constant diameter overlay network for publish/subscribe communication with many topics. We present three overlay network construction heuristics that guarantees constant diameter and topic-connectivity which are most important factors for efficient routing. The formal problem is as follows:

Constant Diameter Topic-Connected Overlay (CD-TCO) Problem: Given a collection of nodes V , a set of topics T , and the node interest assignment I , connect the nodes in V into a topic-connected overlay network G which has least possible average degree and constant diameter.

This problem aims at minimizing the average degree, as does the MinAv-TCO problem introduced by Chockler et. al. [Chockler et al., 2007a], with the additional requirement on diameter. We present three heuristics for this problem and validate our heuristics via experimental results.

We first show that the GM algorithm does not work well for the CD-TCO problem. Second, we present our 2-diameter algorithms and their performance evaluation.

4.4.1. The GM Algorithm and the CD-TCO Problem

The GM algorithm does not work well for the CD-TCO problem: The diameter obtained by the GM algorithm may be as bad as $\Theta(n)$, as we show in the lemma below.

Lemma 4.4.1 *The GM algorithm can have diameter of $\Theta(n)$, where n is number of nodes in the pub/sub system.*

Proof. Consider the example where we have n nodes $v_1, v_2, \dots, v_n$, one topic t and every node is interested in t . There exists many orderings of the edges of G for which the GM algorithm would produce an overlay network with diameter $n - 1$. For example, the overlay network, $G(V, E)$, where $E = \{(v_i, v_{i+1}) | 1 \leq i < n\}$, can result from the GM algorithm – the diameter of this overlay network is $n - 1$. Another solution for the CD-TCO on the same configuration for the nodes V is the overlay network $G(V, E')$, where $E' = \{(v_1, v_i) | 1 < i \leq n\}$, which has diameter 2.

□

4.4.2. Examples I and II

In Figure 17(a), each node is subscribed to topics 1 and 2. In an optimal solution, we have $n - 1$ edges and we have only one star. From this example, we can have the intuition that we should have minimum number of stars.

Intuition 4.4.2 *Construct minimum number of stars.*

In Figure 17(b), node u is subscribed to topics 1, 2, 3, 4 and 5. $(n - 1)/4$ nodes are subscribed to topics 1 and 2; $(n - 1)/4$ nodes are subscribed to topics 2 and 3; $(n - 1)/4$

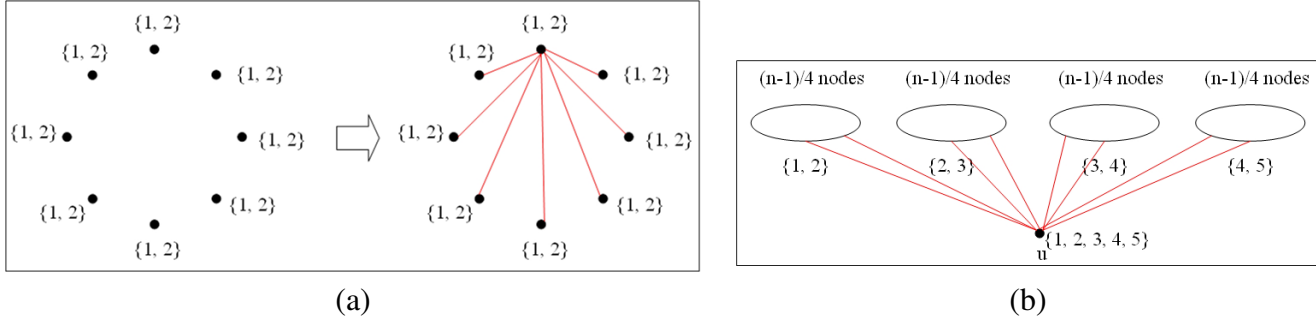


Figure 17. (a) Example I (b) Example II

nodes are subscribed to topics 3 and 4; $(n - 1)/4$ nodes are subscribed to topics 4 and 5. In the optimal solution, we have $n - 1$ edges and we have only one star with center node u . From this example, we can have the intuition that nodes with many neighbors are good candidates for center of stars.

Intuition 4.4.3 *Nodes with many neighbors are good candidates for center of stars.*

4.4.3. Constant Diameter Overlay Design Algorithm (CD-ODA)

CD-ODA starts with the overlay network $G(V, \emptyset)$. At each iteration of the CD-ODA, a node which has maximum number neighbors with non-empty interest intersection is chosen. The number of neighbors of a node u is equal to

$$n_u = |\{v \in V | \exists t \in T, Int(v, t) = Int(u, t) = 1\}|$$

We then put an edge between this node and each of its neighbors, and remove all the topics in this node's interest assignment from the set of topics.

CD-ODA finds the optimal solution for Examples I and II in Figure 17.

Algorithm 13 Constant Diameter Overlay Design Algorithm (CD-ODA)

- 1: $T \leftarrow$ Set of all topics
 - 2: **while** T is not empty **do**
 - 3: For each node u , calculate number of nodes v such that there exists a topic t in T and $Int(u, t) = Int(v, t) = 1$. Denote this number by n_u .
 - 4: Find node u with maximum n_u .
 - 5: Put an edge between u and all nodes v such that there exists a topic $t \in T$ and $Int(u, t) = Int(v, t) = 1$.
 - 6: Remove all topics t from T such that $Int(u, t) = 1$.
 - 7: **end while**
-

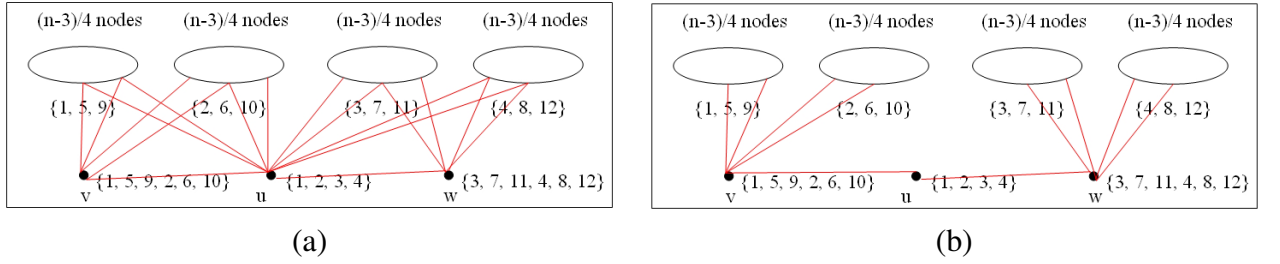


Figure 18. (a) CD-ODA on Example III (b) Optimal solution for Example III

4.4.4. Example III

In Figure 18(a), node u is subscribed to topics 1, 2, 3 and 4; node v is subscribed to topics 1, 5, 9, 2, 6 and 10; and node w is subscribed to topics 3, 7, 11, 4, 8 and 12. $(n - 3)/4$ nodes are subscribed to topics 1, 5 and 9; $(n - 3)/4$ nodes are subscribed to topics 2, 6 and 10; $(n - 3)/4$ nodes are subscribed to topics 3, 7 and 11; $(n - 3)/4$ nodes are subscribed to topics 4, 8 and 12. For this example, CD-ODA first puts edges between node u and all the other nodes and then it puts edges between node v and first two sets of $(n - 3)/4$ nodes and it puts edges between node w and last two sets of $(n - 3)/4$ nodes. Thus, CD-ODA uses $(n - 1) + (n - 3)/2 + (n - 3)/2 = (2n - 4)$ edges.

In the optimal solution (Figure 18(b)), there are edges between node v and first two sets of $(n - 3)/4$ nodes and there are edges between node w and last two sets of $(n - 3)/4$

nodes and there are edges (u, v) and (u, w) . In the optimal solution, only $(n - 3)/2 + (n - 3)/2 + 2 = (n - 1)$ edges are required. From this example, we can have the intuition that nodes with many weighted neighbors are good candidates for center of stars.

Intuition 4.4.4 *Nodes with many weighted neighbors are good candidates for center of stars.*

With using the examples I, II and III in Figures 17(a), 17(b) and 18(b) and intuitions 4.4.2, 4.4.3 and 4.4.4, we designed CD-ODA-I algorithm. CD-ODA-I finds the optimal solution for Example III in Figure 18(b).

4.4.5. Constant Diameter Overlay Design Algorithm I(CD-ODA-I)

The second heuristic presented, the *Constant Diameter Overlay Design Algorithm I (CD-ODA-I)*, starts with the overlay network $G(V, \emptyset)$. At each iteration of the CD-ODA-I, a node u which has maximum number of weighted neighbors is chosen. The number of weighted neighbors of a node z is equal to

$$w_z = \sum_{t \in T} |\{v \in V | Int(v, t) = Int(z, t) = 1\}|.$$

We add an edge between u and each of its neighbors, removing the topics in this node's interest assignment from the set of topics.

4.4.6. Example IV

In Figure 19(a), node u is subscribed to topics $x_1, x_2, \dots, x_k$. $(n - 1)/(2k)$ nodes (set B_1) are subscribed to y_1 , $(n - 1)/(2k)$ nodes (set B_2) are subscribed to $y_2, \dots, (n - 1)/(2k)$

Algorithm 14 Constant Diameter Overlay Design Algorithm I(CD-ODA-I)

- 1: $T \leftarrow$ Set of all topics
 - 2: **while** T is not empty **do**
 - 3: For each node u , calculate total number of weighted neighbors. Denote this number by w_u .
 - 4: Find node u with maximum w_u .
 - 5: Put an edge between u and all nodes v such that there exists a topic $t \in T$ and $Int(u, t) = Int(v, t) = 1$.
 - 6: Remove all topics t from T such that $Int(u, t) = 1$.
 - 7: **end while**
-

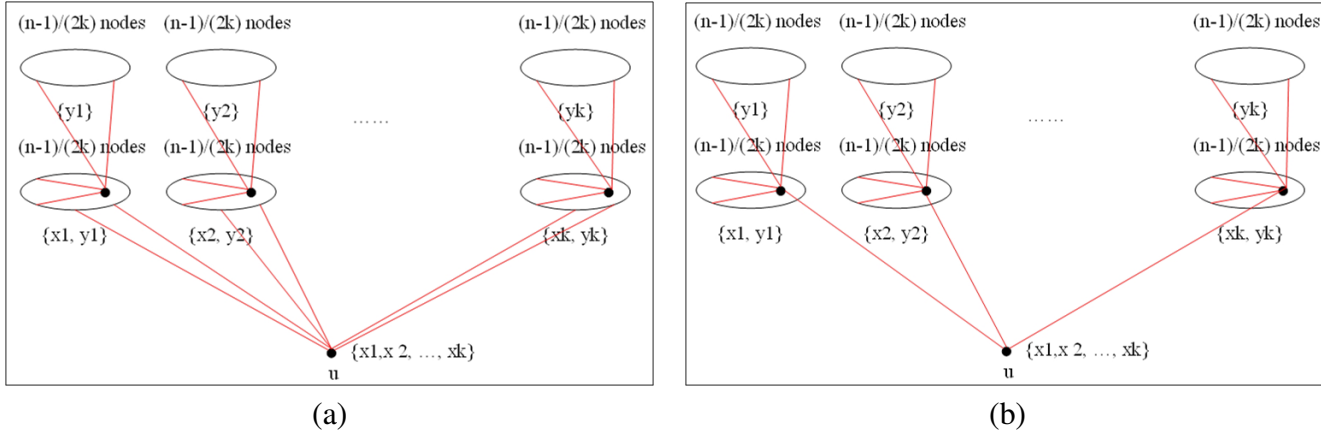


Figure 19. (a) CD-ODA and CD-ODA-I on Example IV (b) Optimal solution for Example IV

nodes (set B_k) are subscribed to y_k . $(n-1)/(2k)$ nodes (set A_1) are subscribed to x_1, y_1 , $(n-1)/(2k)$ nodes (set A_2) are subscribed to $x_2, y_2, \dots$, $(n-1)/(2k)$ nodes (set A_k) are subscribed to x_k, y_k . For this example, CD-ODA and CD-ODA-I first put edges between node u and all the nodes in sets $A_1, A_2, \dots, A_k$ and then it puts edges between nodes $u_i \in A_i$ and all nodes in A_i and B_i , for $1 \leq i \leq k$. Thus, CD-ODA and CD-ODA-I use $((n-1)/(2k)) * k + ((n-1)/k - 1) * k = 3 * (n-1)/2 - k$ edges.

In the optimal solution (Figure 19(b)), there are edges between nodes $u_i \in A_i$ and all nodes in A_i and B_i , for $1 \leq i \leq k$. And there are edges between node u and nodes u_i ,

for $1 \leq i \leq k$. In the optimal solution, only $(k + ((n - 1)/k - 1) * k = (n - 1)$ edges are required. From this example, we can have the intuition that nodes with dense connections are good candidates for center of stars.

Intuition 4.4.5 *Nodes with dense connections are good candidates for center of stars.*

4.4.7. Constant Diameter Overlay Design Algorithm II(CD-ODA-II)

The third heuristic presented, *Constant Diameter Overlay Design Algorithm II (CD-ODA-II)*, also starts with the overlay network $G(V, \emptyset)$. At each iteration of the CD-ODA-II, a node u which has maximum connection density, d_u , is chosen. The connection density of a node u , d_u , is given by

$$d_u = \frac{\sum_{t \in T} |\{v \in V | Int(v, t) = Int(u, t) = 1\}|}{|\{v \in V | \exists t \in T, Int(v, t) = Int(u, t) = 1\}|}.$$

Note that $d_u = \frac{w_u}{n_u}$. We add an edge between a node u with maximum density and each of its neighbors, removing the topics in this node's interest assignment from the set of topics. (See Figure 15.)

Algorithm 15 Constant Diameter Overlay Design Algorithm II(CD-ODA-II)

- 1: $T \leftarrow$ Set of all topics
 - 2: **while** T is not empty **do**
 - 3: For each node u , calculate its connection density. Denote this number by d_u .
 - 4: Find a node u with maximum d_u .
 - 5: Put an edge between u and all nodes v such that there exists a topic $t \in T$ and $Int(u, t) = Int(v, t) = 1$.
 - 6: Remove all topics t from T such that $Int(u, t) = 1$.
 - 7: **end while**
-

With using the example in Figure 19(a) and intuition 4.4.5, we designed the CD-ODA-II algorithm. CD-ODA-II finds the optimal solution for Example IV in Figure 19(b).

4.4.8. Analysis of Algorithms

Lemma 4.4.6 *CD-ODA terminates within $O(|V|^2 * |T|)$ time.*

Lemma 4.4.7 *CD-ODA generates a 2-diameter overlay for each topic.*

Proof. Since the algorithm generates a star for each topic, each topic overlay network will have diameter 2.

□

4.4.9. Conjectures

Conjecture 4.4.8 *All three algorithms approximate the constant diameter overlay design problem within a logarithmic factor.*

Our first intuition is that if there exists a k -edge overlay network with constant diameter for a graph G , then there exists a constant c such that there exists a $c * k$ -edge overlay with diameter 2 for a graph G . This step will make the reduction from constant diameter overlay to 2-diameter overlay.

For 2-diameter overlay, the most efficient graph for each topic alone will have a star structure, since a star has diameter 2 and optimal number of edges ($n-1$ edges). Combining these two steps, we have Conjecture 1.

Conjecture 4.4.9 *There exists no polynomial time algorithm that approximates constant diameter overlay design problem with a constant factor unless $P=NP$.*

Our intuition for Conjecture 2 is that the set cover problem can be reduced to this problem as for the MinMax-TCO and Min-TCO problems.

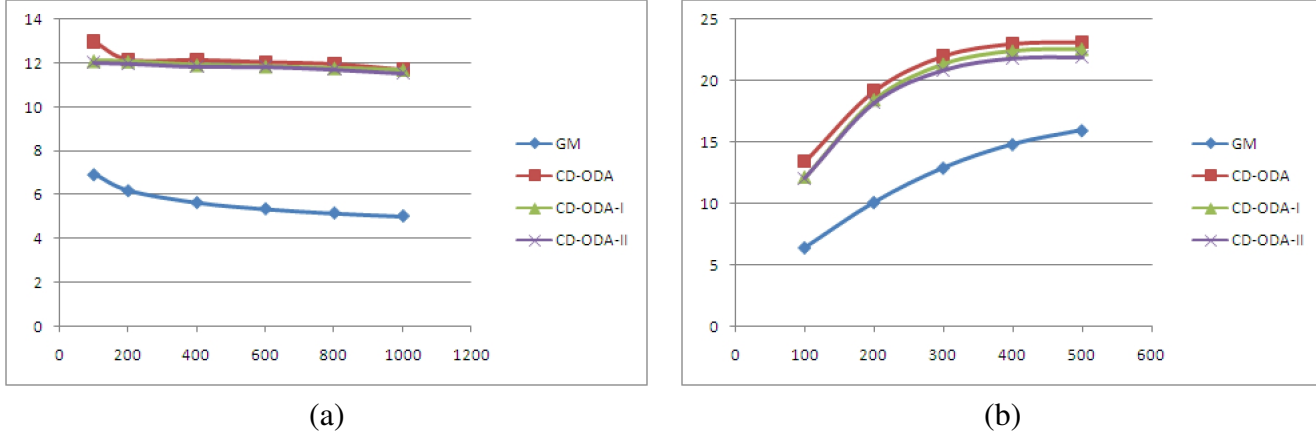


Figure 20. Average node degree for GM, CD-ODA, CD-ODA-I and CD-ODA-II

4.4.10. Experimental Results

The GM algorithm [Chockler et al., 2007a] and our algorithms are implemented in Java. These algorithms are compared according to the average degree in the resulting graph. The diameter is always 2 for our algorithms and for CD-ODA and it may be $\theta(n)$, n : number of nodes, for GM algorithm. When we compare the results of GM and our algorithms according to average degree, our algorithms require at most 2.3 times more edges than GM. Our algorithms improve diameter drastically.

4.4.10.1. AVERAGE NODE DEGREE WITH VARYING NUMBER OF NODES

For the first experiment, the number of nodes varies between 100 to 1000. The number of topics is 100. We fixed number of subscriptions to $s = 10$. Each node is interested in each topic uniformly at random. This experimental setting is similar to previous studies [Chockler et al., 2007a, Ramasubramanian et al., 2006, Onus and Richa, 2009b].

Figure 20(a) is a comparison of GM and our algorithms according to the average degree. The average degree of the graph decreases for GM algorithm when the number of

nodes increases since GM algorithm can find edges with higher correlation as the number of nodes increases. The average degree of the graph slightly decreases for our algorithms. When we compare the results of GM and our algorithm CD-ODA-II, our algorithm requires at most 2.3 times more edges than GM (Figure 20(a)). CD-ODA-II improves CD-ODA by a factor of 3% on average and CD-ODA-I by a factor of 2% on average.

4.4.10.2. AVERAGE NODE DEGREE WITH VARYING NUMBER OF TOPICS

For the second experiment, the number of nodes is 100. The number of topics varies between 100 and 500. We fixed number of subscriptions to $s = 20$. Each node is interested in each topic uniformly at random. This experimental setting is similar to previous studies [Chockler et al., 2007a, Ramasubramanian et al., 2006].

Figure 20(b) is a comparison of GM and our algorithms according to the average degree. The average degree of the graph increases for our algorithms and GM algorithm when the number of topics increases since algorithms can only find edges with lower correlation as the number of topics increases. When we compare the results of GM and our algorithm CD-ODA-II, our algorithm requires at most 1.9 times more edges than GM (Figure 20(b)). CD-ODA-II improves CD-ODA by factor 6% on average and CD-ODA-I by factor 2% on average.

4.4.10.3. AVERAGE NODE DEGREE WITH VARYING SUBSCRIPTION SIZE

For the third experiment, the number of nodes and the number of topics are fixed to 100. The subscription size varies between 10 to 50. Each node is interested in each topic uniformly randomly. This experimental setting is similar to previous studies

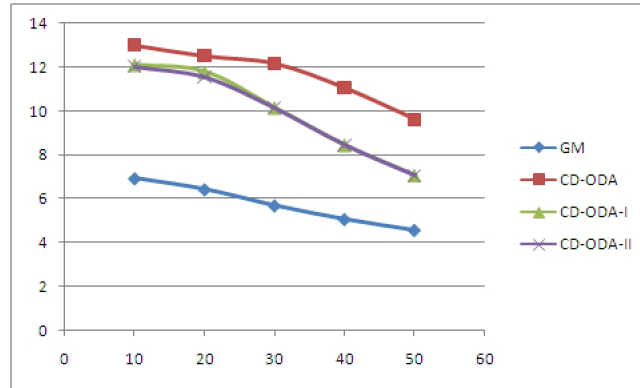


Figure 21. Average node degree for GM, CD-ODA, CD-ODA-I and CD-ODA-II

[Chockler et al., 2007a, Ramasubramanian et al., 2006].

Figure 21 is a comparison of GM and our algorithms according to the average degree. The average degree of the overlay network decreases for both GM and our algorithms when subscription size increases since algorithms can find edges with higher correlation. When we compare the results of GM and our algorithm CD-ODA-II, our algorithm requires at most 1.8 times more edges than GM (Figure 21). CD-ODA-II improves CD-ODA by factor 20% on average and CD-ODA-I by factor 1% on average.

4.5. Conclusion

In Section 4.2, we study a new optimization problem (MinMax-TCO) that constructs a practical and scalable overlay network for publish/subscribe communication with many topics. We present a topic-connected overlay network design algorithm (MinMax-ODA) which approximates the MinMax-TCO problem within a logarithmic factor. We also show that the approximation factor of MinMax-ODA is almost tight, since no constant-

approximation polynomial-time algorithm can exist for the MinMax-TCO problem (unless $P=NP$). Our experimental results validate our formal analysis of the MinMax-ODA algorithm, showing that the maximum degree obtained by our algorithm clearly outperforms the maximum degree obtained when GM is used.

In Section 4.3, we study a new optimization problem (Low-TCO) that constructs a practical and scalable overlay network for publish/subscribe communication with many topics. We present a topic-connected overlay network design algorithm (Low-ODA) which approximates both average degree and maximum degree well. We anticipate that the parameterized algorithmic framework proposed by Low-ODA will be applicable in other network design domains where, for scalability, it is important to keep both the maximum degree as well as the average degree of an overlay network low.

In Section 4.4, we present three heuristics for constructing constant diameter overlay networks. Our experimental results show that the diameter obtained by our heuristics outperforms the diameter obtained when GM is used while only increasing the average degree by a factor of 2.3.

REFERENCES

- [Adler and Scheideler, 2000] Adler, M. and Scheideler, C. Efficient communication strategies for ad hoc wireless networks. *Theory of Computing Systems*, 33:337–391, 2000.
- [Alzoubi et al., 2003] Alzoubi, K., Li, X., Wang, Y., Wan, P., and Frieder, O. Geometric spanners for wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(4), 2003.
- [Alzoubi et al., 2002a] Alzoubi, K., Wan, P.-J., and Frieder, O. New distributed algorithm for connected dominating set in wireless ad hoc networks. In *Proc. of the IEEE Hawaii International Conference on System Sciences (HICSS)*, 2002.
- [Alzoubi et al., 2002b] Alzoubi, K. M., Wan, P.-J., and Frieder, O. Distributed construction of connected dominating set in wireless ad hoc networks. In *Proc. of the Infocom*, 2002.
- [Anceaume et al., 2006] Anceaume, E., Gradinariu, M., Datta, A. K., Simon, G., and Virgillito, A. A semantic overlay for self- peer-to-peer publish/subscribe. In *ICDCS*, page 22, 2006.
- [Angluin et al., 2005] Angluin, D., Aspnes, J., Chen, J., Wu, Y., and Yin, Y. Fast construction of overlay networks. In *SPAA*, pages 145–154, 2005.
- [Aspnes and Shah, 2003] Aspnes, J. and Shah, G. Skip graphs. In *SODA*, pages 384–393, 2003.
- [Awerbuch and Scheideler, 2004] Awerbuch, B. and Scheideler, C. Group spreading: A protocol for provably secure distributed name service. In *ICALP*, pages 183–195, 2004.
- [Baehni et al., 2004] Baehni, S., Eugster, P. T., and Guerraoui, R. Data-aware multicast. In *DSN*, pages 233–242, 2004.
- [Baldoni et al., 2007a] Baldoni, R., Beraldi, R., Quéma, V., Querzoni, L., and Piergiovanni, S. T. Tera: topic-based event routing for peer-to-peer architectures. In *DEBS*, pages 2–13, 2007.
- [Baldoni et al., 2007b] Baldoni, R., Beraldi, R., Querzoni, L., and Virgillito, A. Efficient publish/subscribe through a self-organizing broker overlay and its application to siena. *Comput. J.*, 50(4):444–459, 2007.

- [Banerjee et al., 2002] Banerjee, S., Bhattacharjee, B., and Kommareddy, C. Scalable application layer multicast. In *SIGCOMM*, pages 205–217, 2002.
- [Barabasi and Albert, 1999] Barabasi, A.-L. and Albert, R. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
- [Bhola et al., 2002] Bhola, S., Strom, R. E., Bagchi, S., Zhao, Y., and Auerbach, J. S. Exactly-once delivery in a content-based publish-subscribe system. In *DSN*, pages 7–16, 2002.
- [Broder et al., 2000] Broder, A. Z., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. L. Graph structure in the web. *Computer Networks*, 33(1-6):309–320, 2000.
- [Brzezinski and Szychowiak, 2000] Brzezinski, J. and Szychowiak, M. Self-stabilization in distributed systems - a short survey, 2000.
- [Carzaniga et al., 2004] Carzaniga, A., Rutherford, M. J., and Wolf, A. L. A routing scheme for content-based networking. In *INFOCOM*, 2004.
- [Castro et al., 2002] Castro, M., Druschel, P., Kermarrec, A. M., and Rowstron, A. Scribe: a large-scale and decentralized application-level multicast infrastructure. *IEEE J. Selected Areas in Comm. (JSAC)*, 20(8):1489–1499, 2002.
- [Chand and Felber, 2005] Chand, R. and Felber, P. Semantic peer-to-peer overlays for publish/subscribe networks. In *Euro-Par*, pages 1194–1204, 2005.
- [Cheng et al., 2003] Cheng, X., Huang, D., Li, W., Wu, W., and Du, D. Z. A polynomial-time approximation scheme for the minimum-connected dominating set in ad hoc wireless networks. *NETWORKS: Networks: An International Journal*, 42, 2003.
- [Chockler et al., 2007a] Chockler, G., Melamed, R., Tock, Y., and Vitenberg, R. Constructing scalable overlays for pub-sub with many topics. In *PODC*, pages 109–118, 2007.
- [Chockler et al., 2007b] Chockler, G., Melamed, R., Tock, Y., and Vitenberg, R. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *DEBS*, pages 14–25, 2007.

- [Choi, 2002] Choi, S. IEEE 802.11e MAC-level FEC performance evaluation and enhancement. In *GLOBECOM '02*, 2002.
- [Clark et al., 1990] Clark, B. N., Colbourn, C. J., and Johnson, D. S. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.
- [Cramer and Fuhrmann, 2005] Cramer, C. and Fuhrmann, T. Self-stabilizing ring networks on connected graphs, 2005.
- [Diestel, 2000] Diestel, R. Graph theory, 2000.
- [Dijkstra, 1974] Dijkstra, E. W. Self stabilization in spite of distributed control. *Communications of the ACM*, 17:643–644, 1974.
- [Dolev, 2000] Dolev, S. Self-stabilization, 2000.
- [Dubhashi et al., 2003] Dubhashi, D., Mei, A., Panconesi, A., Radhakrishnan, J., and Srinivasan, A. Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons. In *Proc. of the ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 717–724, 2003.
- [Enachescu et al., 2004] Enachescu, M., Goel, A., Govindan, R., and Motwani, R. Scale free aggregation in sensor networks. In *First Intl. W. on Algorithmic Aspects of Wireless Sensor Networks*, 2004.
- [Eugster et al., 2003] Eugster, P. T., Felber, P., Guerraoui, R., and Kermarrec, A.-M. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.
- [F. Chung, 2002] F. Chung, L. L. Connected components in random graphs with given degree sequences. *Annals of Combinatorics*, 6:125–145, 2002.
- [Faloutsos et al., 1999] Faloutsos, M., Faloutsos, P., and Faloutsos, C. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.
- [Gabriel and Sokal, 1969] Gabriel, K. and Sokal, R. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

- [Gao et al., 2001] Gao, J., Guibas, L., Hershberger, J., Zhang, L., and Zhu, A. Geometric spanner for routing in mobile networks. In *Proc. of the the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 45–55, 2001.
- [Gupta and Kumar, 2000] Gupta, P. and Kumar, P. The capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46(2):388–404, 2000.
- [Gupta and Kumar, 2001a] Gupta, P. and Kumar, P. Internets in the sky: The capacity of three dimensional wireless networks. *Communications in Information and Systems*, 1(1):33–50, 2001.
- [Gupta and Kumar, 2001b] Gupta, P. and Kumar, P. Towards an information theory of large networks: An achievable rate region. In *IEEE International Symposium on Information Theory (ISIT)*, 2001.
- [Herman, 2002] Herman, T. Self-stabilization bibliography: Access guide, 2002.
- [Huang et al., 2004] Huang, H., Richa, A. W., and Segal, M. Approximation algorithms for the mobile piercing set problem with applications to clustering in ad-hoc networks. *ACM Baltzer Journal on Mobile Networks and Applications (MONET)*, pages 141–149, 2004.
- [Intanagonwiwat et al., 2000] Intanagonwiwat, C., Govindan, R., and Estrin, D. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Mobile Computing and Networking*, pages 56–67, 2000.
- [Jia et al., 2001] Jia, L., Rajaraman, R., and Suel, T. An efficient distributed algorithm for constructing small dominating sets. In *Proc. of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2001.
- [Jurdzinski and Stachowiak, 2002] Jurdzinski, T. and Stachowiak, G. Probabilistic algorithms for the wakeup problem in single-hop radio networks. In *Proc. of the International Symposium on Algorithms and Computation (ISSAC)*, pages 535–549, 2002.
- [Kothapalli, 2006] Kothapalli, K. Topology and routing in overlay networks. In *PhD Thesis, Johns Hopkins University, 2006.*, 2006.

- [Kothapalli et al., 2005] Kothapalli, K., Onus, M., Richa, A., and Scheideler, C. Constant density spanners for wireless ad-hoc networks. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 116–125, 2005.
- [Kuhn et al., 2004] Kuhn, F., Moscibroda, T., and Wattenhofer, R. Radio network clustering from scratch. In *Proc. of the European Symposium on Algorithms (ESA)*, 2004.
- [Kuhn and Wattenhofer, 2003] Kuhn, F. and Wattenhofer, R. Constant-time distributed dominating set approximation. In *Proc. of the ACM Symposium on Principles of Distributed Computing (PODC)*, pages 25–32, 2003.
- [Kuhn et al., 2003a] Kuhn, F., Wattenhofer, R., Zhang, Y., and Zollinger, A. Geometric ad-hoc routing: of theory and practice. In *Proc. of the ACM Symposium on Principles of Distributed Computing (PODC)*, 2003.
- [Kuhn et al., 2003b] Kuhn, F., Wattenhofer, R., and Zollinger, A. Ad-hoc networks beyond unit disk graphs. In *Proc. of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 69–78, 2003.
- [Kuruvila et al., 2004] Kuruvila, J., Nayak, A., and Stojmenovic, I. Hop count optimal position based packet routing algorithms for ad hoc wireless networks with a realistic physical layer. In *1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2004.
- [Lau et al., 2007] Lau, L. C., Naor, J., Salavatipour, M. R., and Singh, M. Survivable network design with degree or order constraints. In *STOC*, pages 651–660, 2007.
- [Levis, 1999] Levis, R. Advanced massaging applications with msmq and mqseries, 1999.
- [Li et al., 2002] Li, X.-Y., Calinescu, G., and Wan, P.-J. Distributed construction of planar spanner and routing for ad hoc wireless networks. In *Proc. of the Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM)*, pages 1268–1277, 2002.
- [Li et al., 2001] Li, X.-Y., Wan, P.-J., and Wang, Y. Power efficient and sparse spanner or wireless ad hoc networks. In *IEEE International Conference on Computer Communications and Networks (ICCCN)*, pages 564–567, 2001.

- [Liu et al., 2005] Liu, H., Ramasubramanian, V., and Sirer, E. G. Client behavior and feed characteristics of rss, a publish-subscribe system for web micronews. In *Internet Measurment Conference*, pages 29–34, 2005.
- [Luby, 1985] Luby, M. A simple parallel algorithm for the maximal independent set problem. In *Proc. of the 17th ACM Symposium on Theory of Computing (STOC)*, pages 1–10, 1985.
- [Onus et al., 2005] Onus, M., Kothapalli, K., Richa, A., and Scheideler, C. Efficient broadcasting and gathering in wireless ad hoc networks. In *IEEE International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN)*, 2005.
- [Onus and Richa, 2009a] Onus, M. and Richa, A. W. Brief announcement: Parameterized maximum and average degree approximation in topic-based publish-subscribe overlay network design. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2009.
- [Onus and Richa, 2009b] Onus, M. and Richa, A. W. Minimum maximum degree publish-subscribe overlay network design. In *INFOCOM*, 2009.
- [Onus et al., 2007] Onus, M., Richa, A. W., and Scheideler, C. Linearization: Locally self-stabilizing sorting in graphs. In *ALLENEX*, 2007.
- [Oracle, 2000] Oracle. Oracle9i application developers guide advanced queuing, 2000.
- [P802.11a/D7.0-1999, 1999] P802.11a/D7.0-1999, I. Wireless LAN medium access control (MAC) and physical layer (PHY) specification: High speed physical layer in the 6 ghz band. IEEE, New York, 1999.
- [Parthasarathy and Gandhi, 2004] Parthasarathy, S. and Gandhi, R. Distributed algorithms for coloring and domination in wireless ad hoc networks. In *Proc. of FSTTCS*, 2004.
- [Quin and Kunz, 2003] Quin, L. and Kunz, T. On-demand routing in MANETs: The impact of a realistic physical layer model. In *Proc. of the 2nd International Conference ADHOC-NOW*, pages 37–48, 2003.
- [Ramasubramanian et al., 2006] Ramasubramanian, V., Peterson, R., and Sirer, E. G. Corona: A high performance publish-subscribe system for the world wide web. In *NSDI*, 2006.

- [Rappaport, 1996] Rappaport, T. *Wireless Communications: Principle and Practice*. Prentice Hall, 1996.
- [Ripeanu et al., 2002] Ripeanu, M., Foster, I. T., and Iamnitchi, A. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *CoRR*, cs.DC/0209028, 2002.
- [Sandler et al., 2005] Sandler, D., Mislove, A., Post, A., and Druschel, P. Feedtree: Sharing web micronews with peer-to-peer event notification. In *IPTPS*, pages 141–151, 2005.
- [Santis et al., 2007] Santis, E. D., Grandoni, F., and Panconesi, A. Fast low degree connectivity of ad-hoc networks via percolation. In *ESA*, pages 206–217, 2007.
- [Shaker and Reeves, 2005] Shaker, A. and Reeves, D. S. Self-stabilizing structured ring topology p2p systems. In *Peer-to-Peer Computing*, pages 39–46, 2005.
- [Song et al., 2004] Song, W.-Z., Wang, Y., Li, X.-Y., and Frieder, O. Localized algorithms for energy efficient topology in wireless ad hoc networks. In *Proc. of the ACM International Symposium on Mobile ad hoc networking and computing (MOBIHOC)*, pages 98–108, 2004.
- [Stoica et al., 2001] Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.
- [Tam et al., 2003] Tam, D., Azimi, R., and Jacobsen, H.-A. Building content-based publish/subscribe systems with distributed hash tables. In *DBISP2P*, pages 138–152, 2003.
- [Tock et al., 2005] Tock, Y., Naaman, N., Harpaz, A., and Gershinsky, G. Hierarchical clustering of message flows in a multicast data dissemination system. In *IASTED PDCS*, pages 320–326, 2005.
- [Voulgaris et al., 2006] Voulgaris, S., Riviere, E., Kermarrec, A. M., and van Steen, M. Sub-2-sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks. In *IPTPS*, 2006.

- [Wang and Li, 2002] Wang, Y. and Li, X.-Y. Geometric spanners for wireless ad hoc networks. In *Proc. of the IEEE International Conference Distributed Computing Systems (ICDCS)*, 2002.
- [Wang and Li, 2003] Wang, Y. and Li, X.-Y. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. In *Proceedings of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 59–68, 2003.
- [Wattenhofer et al., 2001] Wattenhofer, R., Li, J., Bahl, P., and Wang, Y.-M. Distributed topology control for wireless multihop ad-hoc networks. In *IEEE INFOCOM '01*, pages 1388–1397, 2001.
- [Yao, 1982] Yao, A. C. C. On constructing minimum spanning trees in k -dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.
- [Yu et al., 2004] Yu, Y., Krishnamachari, B., and Prasanna, V. Energy-latency tradeoffs for data gathering in wireless sensor networks. In *IEEE Infocom*, 2004.