OPTIMIZATION ALGORITHMS FOR INFORMATION RETRIEVAL AND TRANSMISSION IN DISTRIBUTED AD HOC NETWORKS

A Dissertation

by

HONG LU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2008

Major Subject: Computer Science

OPTIMIZATION ALGORITHMS FOR INFORMATION RETRIEVAL AND TRANSMISSION IN DISTRIBUTED AD HOC NETWORKS

A Dissertation

by

HONG LU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Jyh Liu
Committee Members,	Donald K. Friesen
	Jianer Chen
	Weiping Shi
Head of Department,	Valerie E. Taylor

December 2008

Major Subject: Computer Science

ABSTRACT

Optimization Algorithms for Information Retrieval and Transmission in Distributed Ad Hoc Networks. (December 2008)

Hong Lu, B.E., Southeast University, China;

M.S., Southeast University, China

Chair of Advisory Committee: Jyh Liu

An ad hoc network is formed by a group of self-configuring nodes, typically deployed in two or three dimensional spaces, and communicating with each other through wireless or some other media. The distinct characteristics of ad hoc networks include the lack of pre-designed infrastructure, the natural correlation between the network topology and geometry, and limited communication and computation resources. These characteristics introduce new challenges and opportunities for designing ad hoc network applications. This dissertation studies various optimization problems in ad hoc network information retrieval and transmission.

Information stored in ad hoc networks is naturally associated with its location. To effectively retrieve such information, we study two fundamental problems, range search and object locating, from a distance sensitive point of view, where the retrieval cost depends on the distance between the user and the target information. We develop a general framework that is applicable to both problems for optimizing the storage overhead while maintaining the distance sensitive retrieval requirement. In addition, we derive a lowerbound result for the object locating problem which shows that logarithmic storage overhead is asymptotically optimal to achieve linear retrieval cost for growth bounded networks.

Bandwidth is a scarce resource for wireless ad hoc networks, and its proper utilization is crucial to effective information transmission. To avoid conflict of wireless

transmissions, links need to be carefully scheduled to satisfy various constraints. In this part of the study, we first consider an optimization problem of end-to-end on-demand bandwidth allocation with the single transceiver constraint. We study its complexity and present a 2-approximation algorithm. We then discuss how to estimate the end-to-end throughput under a widely adopted model for radio signal interference. A method based on identifying certain clique patterns is proposed and shown to have good practical performance.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

A. Ad hoc networks

An ad hoc network is created by a collection of autonomous nodes in a spontaneous manner. Such networks may be rapidly deployed as needed, to establish survivable, efficient, and dynamic communication for many applications such as emergency operations, disaster relief efforts, and etc, where centralized and organized connectivity is not affordable or appropriate. Other application scenarios include battlefield surveillance, environment and habitat monitoring, traffic control, mobile target tracking, home automation, health care applications, and etc [2, 3, 4].

Ad hoc networks differ from other types of networks such as Internet or cellular networks in many ways. First, there is no pre-existing infrastructure in ad hoc networks and the connection between nodes is established on-the-fly. The nodes in ad hoc networks act as both end hosts and routers at the same time, and they typically work in a peer-to-peer fashion. Second, some ad hoc networks consist of small nodes with limited computation and communication resources. They are typically equipped with low cost processors with limited amount of memory and power supply. They normally share a common bandwidth which is a scarce system resource. Third, ad hoc networks are usually deployed in two or three dimensional spaces, and their topologies are closely tied with the underlying geometric environment. Fourth, the data produced in ad hoc networks is often correlated both in the temporal and spatial domain. This kind of data correlation introduces new requirements and opportunities in various applications, ranging from information gathering, aggregation, diffusion,

---

This dissertation follows the style of *IEEE Transactions on Computers*.

and etc.

These characteristics introduce new challenges and opportunities for designing ad hoc network information services. In this dissertation, we study ad hoc networking from the following two perspectives:

- Distance sensitive information retrieval. In particular, we study two problems, range search and object locating .
- End-to-end information transmission, including on-demand bandwidth allocation and throughput estimation.

In this dissertation, these problems are investigated from a technology independent point of view so that the results will remain valid when technologies change. In the rest of this chapter, we first give an overview of these two topics.

B. Distance sensitive information retrieval

One of the major challenges in distributed ad hoc networks is to efficiently retrieve the various information stored in the network. This is particularly true in sensor networks, whose main purpose is to collect and process environmental information. In the study of information retrieval schemes, we are particularly interested in ad hoc networks that are deployed in two dimensional spaces. The topology of such networks usually has a distinct geometric structure and information stored in such networks is naturally associated with its location. In particular, we focus on two fundamental information retrieval problems, range search and object locating. The goal of a range search scheme is to answer range queries posed by users. A range query asks for the information in a region of the network under consideration. For example, Figure 1 shows a two dimensional sensor network, where a user at the node

q asks for the average temperature readings of all the sensor nodes within distance r from himself. Instead of retrieving regional information, object locating concerns searching individual objects. An object location query asks for the location (or the address) of the node who owns an object with a particular identifier.

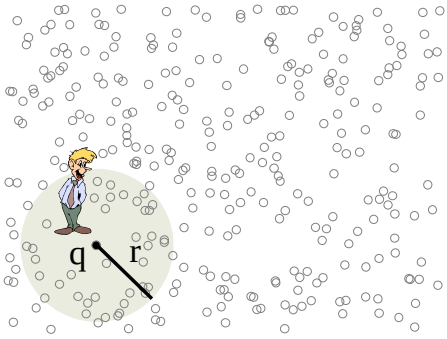


Fig. 1. A distributed range query initiated by the node q in a sensor network. The query asks for the average temperature readings of all the sensor nodes in a circular region with radius r.

An important property in the design of distributed geometric information retrieval systems is distance sensitivity. Roughly speaking, a distance sensitive information retrieval system is one who retrieves local information more efficiently than remote information. Specifically, in a distance sensitive range search system, the cost to retrieve the information in a region increases as the radius of that region, and the relationship is described by a monotonic function specified as a requirement of the system design. For example, one may require that the cost to retrieve the information in a region must not exceed the radius of that region times a constant factor. In a distance sensitive object locating system, the cost to locate an object depends on the distance between the user and the target object. Distance sensitivity is a desirable property mainly because in many ad hoc network applications, local

information is much more valuable than remote information, and therefore, systems that can retrieval local information more efficiently are preferred. Not only does a distance sensitive scheme retrieves local information more efficiently, it also does that in a more robust manner in the following sense. In our discussion the retrieval cost is measured as the length of the retrieval path. For a distance sensitive scheme, the retrieval path for local information is shorter than remote information, and therefore is less likely to be broken in case of network partition.

The main challenge in designing distance sensitive information retrieval schemes is to enforce the distance sensitivity constraint with minimum resource overhead, in particular, minimum storage cost. Achieving the objective of distance sensitive retrieval alone is not difficult. As an extreme case, in the example of Figure 1, if every sensor holds a copy of everybody else's temperature data, then the retrieval cost for any range query is zero in terms of the communication overhead incurred when the system answers the query, which is distance sensitive by definition. However, the high storage cost for the sensor nodes makes this simple scheme unsatisfactory in most cases. The example shows that it is not straightforward at all to maintain the distance sensitivity property of the retrieval process while minimizing the storage cost. In general, one can trade storage overhead for better retrieval performance, and studying this intriguing tradeoff is at the heart of our later discussion on this topic.

In Chapter II, we will discuss the distance sensitive information retrieval problems in more details. Our primary results are twofold. First, we developed a framework that optimizes storage overhead for an arbitrary retrieval cost function for grid networks. This framework is applicable to both the range search problem and the object locating problem. Compared with our method, almost all existing design focus on a special case where the retrieval cost function is a linear one. The performance of our proposed scheme is verified by both theoretical analysis and simulation evaluation.

Second, we derived a lowerbound result which shows that logarithmic storage cost is asymptotically optimal to achieve linear retrieval cost for a large class of networks called the growth bounded networks. This result proves the optimality of our object locating schemes, as well as many existing ones in the literature. In addition to these two main results, we also address other relevant issues such as load balancing, irregular network topology, and etc.

C. End-to-end information transmission

A large number of ad hoc networks are wireless networks. In such networks, bandwidth is usually a scarce resource mainly because it has to be shared by multiple nodes. Efficient utilization of the valuable bandwidth resource, therefore, is of great importance to wireless ad hoc network applications.

An important characteristic of wireless networks is that multiple wireless transmissions may conflict with each other due to various reasons such as radio interference. As a result, wireless transmissions usually need to be carefully scheduled, according to certain schedulability constraints, to avoid such conflicts. This need of transmission scheduling adds considerable complications to the tasks in wireless ad hoc networks. In Chapter III, we study two specific end-to-end information transmission problems, on-demand bandwidth allocation and throughput estimation, with schedulability constraints.

In on-demand bandwidth allocation, we study the following problem: when a user's end-to-end connection request with a specific bandwidth requirement arrives, find a path between the source and destination in question and allocates the bandwidth along this path, such that, (1) the bandwidth requirement of the connection request is satisfied, and (2) the total bandwidth consumption is minimized so that

more future requests can be satisfied. In other words, the objective is to provide services with guaranteed quality at the minimum cost of system resource.

The major challenge in designing efficient bandwidth allocation schemes for wireless ad hoc networks is that it is a simultaneous routing and scheduling problem. Traditional bandwidth aware routing algorithms based on minimizing hop counts are not suitable because they are designed for networks where links are physically isolated. In our problem, however, the wireless links are not independent from each other due to the single transceiver constraint, which says that a set of transmissions can successfully occur at the same time only if no two transmissions share a common sender or receiver. The impact of single transceiver constraint to bandwidth allocation is the focus of our study. In particular, we show that under the single transceiver constraint, the end-to-end on-demand bandwidth allocation problem is NP-hard. Based on this complexity result, we develop a greedy 2-approximation algorithm. In addition to the above theoretical results, a set of simulations are conducted to evaluate the performance of the approximation algorithm in practice. The experiments show that the proposed algorithm significantly outperforms traditional algorithms.

The end-to-end throughput is the maximum amount of data that can be successfully transmitted from a source node to a sink in a given period of time. End-to-end throughput is a system parameter that is very important to many network management tasks such as capacity planning, bottleneck identification, and etc. The end-to-end throughput estimation problem studied in this dissertation is a variant of the well-known maxflow problem with an extra schedulability constraint which models the effect of radio signal interference. This problem has been shown to be NP-hard, and the main contribution of our work is an efficient method with practically good estimation based on identifying cliques in the interference pattern.

CHAPTER II

DISTANCE SENSITIVE INFORMATION RETRIEVAL

A. Introduction

In this chapter, we mainly study two information retrieval problems: range search and object locating in distributed ad hoc networks. A range query asks for all the data or some statistics of the data in a region of the network. For example, a sensor node may ask for the average temperature readings of all the sensor nodes within five meters of itself. An object query asks for the address of the node where an object is located, so that the content of the object can be retrieved later on using the obtained node address. For example, a node may ask the position of the node which has a particular video clip.

A desired property of an information retrieval system for distributed ad hoc networks is distance sensitivity. For range search, distance sensitivity means that the cost to answer a range query should be bounded by a function of the query radius. For object locating, distance sensitivity means that the cost to locate an object must be bounded by a function of the distance between the object and the node which initiates the query.

In this chapter, we will first rigorously formulate the above distance sensitive query processing problems, and then present our design in the context of grid networks. The primary results of this chapter are twofold. First, a framework is developed for designing schemes with an arbitrary retrieval cost function and optimized storage overhead. Second, a lowerbound result is derived for object locating, which shows that logarithmic storage cost is asymptotically optimal to achieve linear retrieval cost for a large class of networks called the growth bounded networks. In

addition to these two main results, we also study other issues in designing distributed distance sensitive information storage and retrieval schemes, such as load balancing, irregular network topology, and etc.

The rest of this chapter is organized as follows. Section B introduces the network model. The range search problem is treated in Section C. Section D studies the object locating problem. The related work is reviewed in Section E. At last, Section F summarizes and concludes this chapter.

B. Network model

In the rest of this chapter, we assume that there is a pre-existing routing layer for the network under consideration. That is, each node is assigned a unique address, and a source can communicate with any other node using the address of the destination. The term distance is used to refer to the routing distance between two nodes, that is, the cost to deliver a unit packet from one to the other.

The distance sensitive information services to be presented are mainly designed for two (or three) dimensional dense ad hoc networks with only short links. Figure 2 shows an example of such networks. We assume that routing is accomplished by a geographic routing algorithm. The geographic routing algorithms [5, 6, 7, 8, 9, 10, 11] developed in the recent years are mostly based on greedy routing augmented by a deadend recovery mechanism. Such routing algorithms have the following two properties. (1) For reasonably dense networks, the routing distance between two nodes is close to the Euclidean distance between them, as long as there is no obstacles in the line of sight of the two nodes. For example, in Figure 2, the length of the s-t path is close to the s-t Euclidean distance. To simplify the discussion, we further assume that the routing distance between any pair of nodes is exactly equal to their

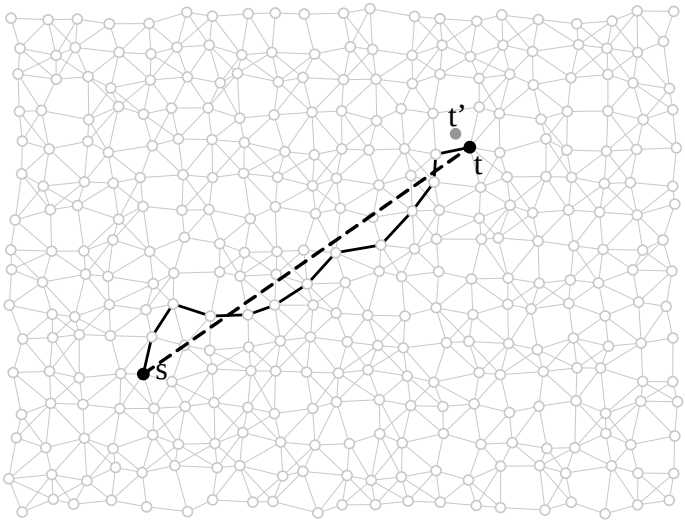


Fig. 2. A dense ad hoc network with only short links. In geographic routing, the routing distance between s and t is roughly equal to their Euclidean distance.

Euclidean distance. Even though this is an unrealistic assumption, it allows us to avoid distinguishing routing and Euclidean distance. And, it will become clear later that this assumption can be easily removed as long as the routing distance is bounded by a constant times the Euclidean distance. (2) A packet destined for a virtual node will be delivered to the closest physical node of the virtual node. Here, a virtual node refers to a node which has a location but does not really exist in the network. For example, in Figure 2, if s tries to deliver a packet to the virtual node t', the packet will be delivered to the node t.

Dense networks with short links are of particular interest because they can be approximated reasonably well by a virtual grid. Figure 3 shows a virtual grid superimposed on top of a physical network. Both of them cover the same area. The design of information retrieval systems can be considerably simplified by mapping each physical node to its nearest virtual grid node. Since it is assumed that the ad hoc network is dense, the Euclidean distance between a physical node and its nearest

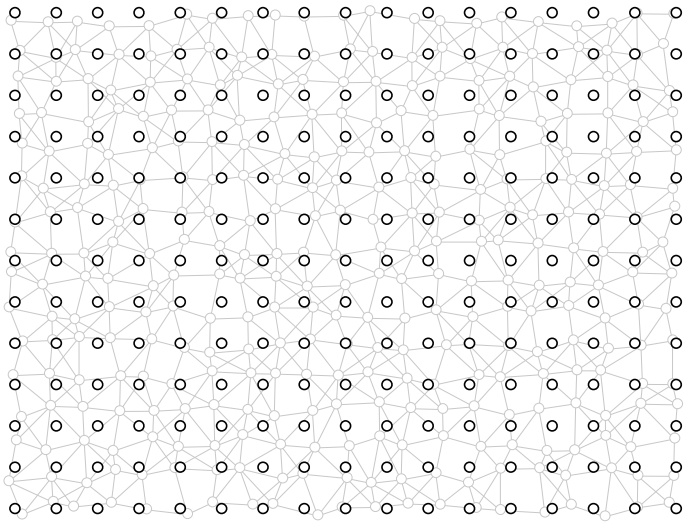


Fig. 3. A virtual grid superimposed on top of a physical ad hoc network in a two dimensional plane with short links.

virtual grid node will be small enough to omit in many applications. Without loss of generality, we assume that the x and y coordinates of each grid node are a pair of integers, and the minimum distance between two grid nodes is one. Even though most of the discussion in this chapter is based on the grid network model, we will show in Section 5 that, the basic principle for designing distance sensitive information services applies to a wider class of networks called growth (upper) bounded networks.

The design of the grid based information retrieval schemes is based on two important notions: d-lattice and d-neighbor. The *d-lattice* nodes (of a grid) are the nodes whose x and y coordinates are both integer multiples of d. The d-neighbor of a node is its closest d-lattice node, where ties are broken arbitrarily. For example, in Figure 4, the 4-lattice nodes are depicted as black dots, and the node q_4 is the 4-neighbor of the node q.

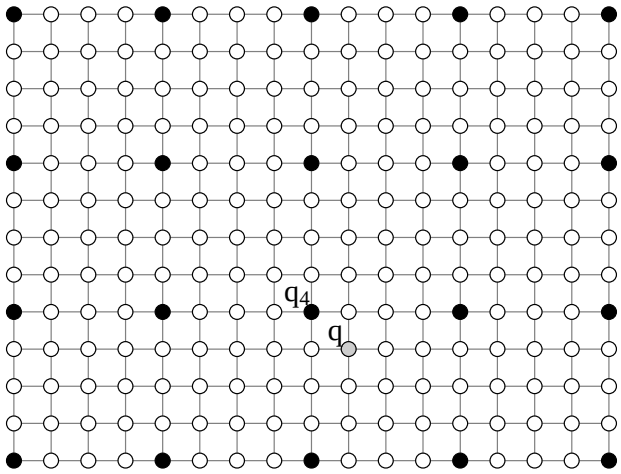


Fig. 4. The concepts of d-lattice and d-neighbor. The 4-lattice nodes are depicted as black dots. q_4 is the 4-neighbor of the node q.

C. Range search

In this Section, we study the distance sensitive range search problem in distributed ad hoc networks. We first formally define the problem in Subsection 1, and then present a baseline design in Subsection 2 with logarithmic storage and linear retrieval cost. The baseline design is generalized in Subsection 3 to support arbitrary retrieval cost function. In Subsection 4, we discuss modifications to the baseline and generic schemes with better load balancing performance.

1. Problem statement

The distributed local range search problem that we study here is a varaint of its well-known centralized version [12]. Consider a set of data distributed across an ad hoc network. Each data item belongs to a node, called its *owner*. A *range query*, initiated by a node in the network, asks for all the data items or some statistics of the data items within a certain distance of the node. In other words, a range query

specifies a circular region centered at a *query initiator*. The data items in the region are referred to as the *target data items*. The owner of a data item may store the data locally, or at one or more other nodes, called its *storage servers* (the owner itself is also a storage server), in order to support efficient data retrieval.

To design a range query scheme, one needs to specify both a storage and a retrieval policy. The storage policy specifies how and where each data item is stored, and a retrieval policy specifies how a range query is answered by the system. There are different ways to measure the performance of a range search design. In our discussion, we choose the following measurement which consists of two components, the storage and retrieval cost. Formally, the *storage cost* of a data item is the number of copies, or the number of storage servers, for that data item. The storage cost of a range search scheme is the maximum storage cost over all data items in the network. The *retrieval cost* is a little more involved, which varies from case to case depending on the retrieval policy. In our discussion, we restrict ourselves to the class of retrieval policies which searches data along a single *retrieval path* starting from the query initiator. The path has the property that every target data item has at least one storage server on it, therefore, by search along the path, one can successfully retrieve all the target data. For such retrieval policies, the length of the retrieval path for a query is called the retrieval cost for that query. Notice that, a query can also be answered by other ways, such as flooding the target region. However, it is beyond the scope of our discussion to study those retrieval policies.

We say a range search scheme is *distance sensitive* if its retrieval policy is distance sensitive. Formally, a range search scheme is distance sensitive with a retrieval function g if the retrieval cost for a query with radius r is at most $g(r)$. Only monotonically increasing functions are considered in our discussion, and therefore, the smaller r is, the smaller the retrieval cost is. Note that this upper bound of the retrieval

cost applies to queries initiated by any node in the network with any query radius r. Distance sensitivity is a desirable property because in many ad hoc network applications, the value of a piece of information to a user is strongly correlated with the distance between them. Usually, local information is much more valuable than remote information, and therefore, it is natural for a system to be able to retrieve local information more efficiently.

Having defined the notion of distance sensitivity, let us look at two simple range search schemes. The first straightforward design is as follows: each data item is stored locally at its owner, and a range query is answered by scoped flooding, that is, by flooding all the nodes within the query radius of the user. The advantage of this local storage design is its low storage cost: there is only one copy for each data item in the network. The obvious disadvantage of this scheme is its high retrieval cost, as flooding is a known communication expensive operation. Another straightforward design is to store each data item at every node such that range queries can be processed locally. Apparently, this design has high storage cost but the retrieval cost in terms of inter-node communication is zero. Both of the above two range search schemes are distance sensitive. They have the opposite performance in terms of storage and retrieval cost. In general, there is an intrinsic tradeoff between the storage and retrieval cost of a distance sensitive scheme, and a significant part of this chapter is spent to discuss this tradeoff.

2. The baseline scheme

The baseline version of the grid based range search scheme is designed to achieve linear retrieval cost. Specifically, the retrieval cost function of the baseline scheme is $g(r) = s \cdot r$, where s is a positive number called the *stretch factor* which can be set by the designer of the scheme.

The baseline scheme adopts a well known technique called exponential storage and retrieval, which was also explored by a number of related papers [13, 14, 15, 16, 17, 18, 19]. Although the baseline scheme differs from them in details, it is not meant to be a radically new design. Rather, it serves as a basis for the discussion of its generic version to be presented in the next subsection.

The baseline scheme is as follows:

- *Storage policy:* a node stores its data at all the 2^i-lattice nodes within distance $d_i + 2^i/\sqrt{2}$, $i = 0, 1, 2, \ldots, m$ of itself, where $m = \lceil \log(s \cdot D/\sqrt{2}) \rceil$.

$$d_i = \sqrt{2} \cdot 2^i/s, \tag{1}$$

  and D is the diameter of the network, i.e., the greatest distance between any two nodes in the network.

- *Retrieval policy:* a node retrieves the target data items at its 2^j-neighbor, where r is the query radius and

$$j = \begin{cases} \lceil \log(s \cdot r/\sqrt{2}) \rceil & \text{if } r > \sqrt{2}/s \\ 0 & \text{if } r \leq \sqrt{2}/s \end{cases}. \tag{2}$$

The above design is illustrated in Figure 5, where q is the initiator of the range query with radius r, p is a generic node within distance r from q, and the storage servers of the node p are depicted as black dots. Now, we analyze the storage and retrieval policy of this design.

According to the storage policy, there are $m = O(\log D)$ layers of storage servers for each data item. In each layer, the number of storage servers is upper bounded by

$$\frac{\pi \cdot ((\sqrt{2}/s) \cdot 2^i + 2^i/\sqrt{2})^2}{(2^i)^2} = \pi \cdot (\sqrt{2}/s + 1/\sqrt{2})^2 = O(1),$$

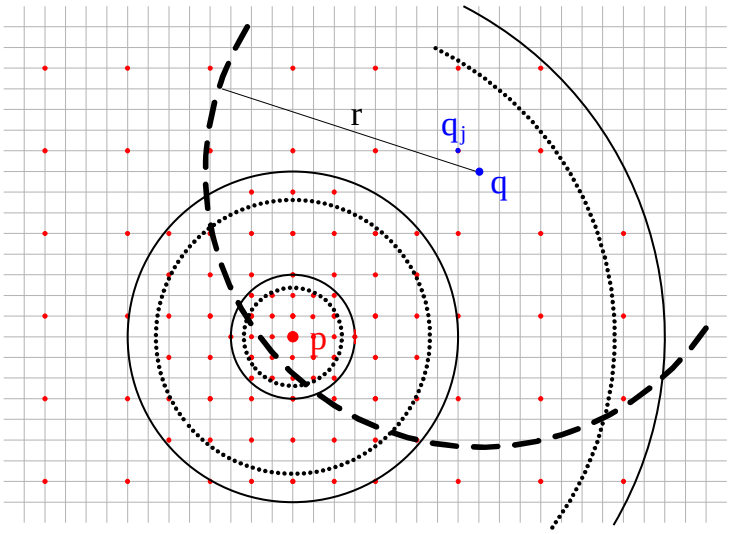


Fig. 5. The baseline range search scheme. q is the query initiator, p is an arbitrary data owner such that $|pq| \leq r$. q_j is the 2^j-neighbor of the node q.

where the numerator of the above equation is the area of the circle centered at the data owner and with radius $d_i + 2^i/\sqrt{2}$, and the denominator is the area of the square with base length 2^i. Therefore, the storage cost for an arbitrary node is $O(\log D)$, which is to say, the storage cost for the baseline scheme is $O(\log D)$.

Now, we show that the retrieval policy is correct, i.e., a node q following the retrieval policy will not miss any target data. Let p be an arbitrary node such that $|pq| \leq r$. We show that q_j, the 2^j-neighbor of q, is a storage server of the node p, or more accurately, any data item owned by p. There are two cases. (a) if $r \leq \sqrt{2}/s$, by (2), $j = 0$ and $d_j = \sqrt{2}/s$, and therefore $|pq| \leq d_j$. The 2^j-neighbor, that is, the 1-neighbor q_1 in this case, is q itself. So $|qq_j| = 0 < 2^j/\sqrt{2}$. (b) if $r > \sqrt{2}/s$, then by (1) and (2),

$$d_j = (\sqrt{2}/s) \cdot 2^{\lceil \log(s \cdot r/\sqrt{2}) \rceil} \geq (\sqrt{2}/s) \cdot 2^{\log(s \cdot r/\sqrt{2})} = r.$$

and thus again $|pq| \leq d_j$. Since q_j is the 2^j-neighbor of q, by the definition of 2^j-

neighbor, $|qq_j| \leq 2^j/\sqrt{2}$. In both of the above two cases, $|pq| \leq d_j$ and $|qq_j| \leq 2^j/\sqrt{2}$, so

$$|pq_j| \leq |pq| + |qq_j| \leq d_j + 2^j/\sqrt{2}.$$

Since r is at most D, j is at most $m = \lceil \log(s \cdot D/\sqrt{2}) \rceil$. According to the storage policy, q_j is one of p's storage servers. Therefore, the retrieval policy is correct.

For this retrieval policy, the retrieval path for a query with center q and radius r is simply qq_j, and the retrieval cost is $|qq_j|$, i.e., the Euclidean distance between q and q_j. When $r > \sqrt{2}/s$, this cost is

$$|qq_j| \leq 2^j/\sqrt{2} = 2^{\lceil \log(s \cdot r/\sqrt{2}) \rceil}/\sqrt{2} < 2^{\log(s \cdot r/\sqrt{2})+1}/\sqrt{2} = s \cdot r.$$

When $r \leq \sqrt{2}/s$, the cost is 0. In both cases, the retrieval cost is at most $s \cdot r$.

The above analysis shows that the baseline scheme achieves linear retrieval at the cost of logarithmic storage, which is summarized as the following theorem.

Theorem 1. *The baseline range search scheme has $O(\log D)$ storage cost and $O(r)$ retrieval cost.*

3. The generic scheme

The previous subsection presents the baseline range search scheme with logarithmic storage and linear retrieval cost. In this subsection, we argue that $O(\log D) + O(r)$ is only one point in the design space, and propose a generalization of the baseline scheme, which has an arbitrary retrieval cost function and an optimized storage cost.

The fundamental strategy of the baseline scheme is to use 2^i-lattice nodes as rendezvous places for data storage and retrieval. The basic idea of the generic scheme is a direct generalization of this strategy, that is, to use l_i-lattice nodes for data storage and retrieval, where l_i is a number sequence different than the exponential

sequence 2^i. Specifically, the generic range search scheme is defined as follows:

- *Storage policy:* a node stores its data at all the l_i-lattice nodes within distance $d_i + l_i/\sqrt{2}$ of itself, $i = 0, 1, 2, \ldots, m$.
- *Retrieval policy:* a node retrieves the target data items at its l_j-neighbor, where j is the smallest non-negative integer such that $d_j \geq r$ and r is the radius of the query.

In the above design, the sequences l_i and d_i are the solution of the following optimization program parameterized by what we call the meta variable m,

$$\min \quad \sum_{i=1}^{m} \pi(d_i + l_i/\sqrt{2})^2/l_i^2 \tag{3}$$

$$s.t. \qquad l_i/\sqrt{2} \leq g(d_{i-1}) \qquad \forall 1 \leq i \leq m \tag{4}$$

$$d_i \geq 0, l_i \geq 0 \qquad \forall 0 \leq i \leq m \tag{5}$$

$$d_0 = 1, d_m \geq D \tag{6}$$

Similar to the baseline scheme, there are m layers of storage servers for each data item. In the ith layer, the number of storage servers is upper bounded by $\pi(d_i + l_i/\sqrt{2})^2/l_i^2$, where the numerator is the area of the circle centered at the data owner and with radius $d_i + 2^i/\sqrt{2}$, and the denominator is the area of the square with base length l_i. Therefore, the objective function (3) of the optimization program is an upperbound of the total storage cost of a data item. The inequality (10) is the non-negativity constraint.

Now we analyze the generic scheme. First, we show that the retrieval policy is correct. Let q be the query initiator, and let p be an arbitrary node within distance r of q, that is, $|pq| \leq r$. Refer to Figure 5 again. By the retrieval policy, $r \leq d_j$. Since $|pq| \leq r$, we have $|pq| \leq d_j$. By the definition of l_j-neighbor, $|qq_j| \leq l_j/\sqrt{2}$, where

q_j is q's l_j-neighbor. Therefore, $|pq_j| \leq |pq| + |qq_j| \leq d_j + l_j/\sqrt{2}$. Since $r < D$, and $d_m \geq D$ by (6), it must be true that $j \leq m$. By the storage policy, q_j must be p's storage server. Therefore, the retrieval policy is correct.

Now we show that the retrieval cost is bounded by $g(r)$. The retrieval cost is $|qq_j|$, and by the definition of l_j-neighbor, $|qq_j| \leq l_j/\sqrt{2}$. By (4), $l_j/\sqrt{2} \leq g(d_{j-1})$. By the retrieval policy, j is the smallest integer such that $r \leq d_j$, therefore it must be true that $d_{j-1} \leq r$. Thus, the retrieval cost is bounded by $g(r)$, which proves the following theorem.

Theorem 2. *The generic range search scheme is distance sensitive with a retrieval function g.*

Constraint (4) is called the *distance sensitivity constraint* in the sense that any l_i and d_i sequence satisfying this constraint plus the storage and retrieval policy yields a distance sensitive range search scheme. In particular, the baseline scheme can be obtained by setting $l_i = 2^i$ and $l_i/\sqrt{2} = s \cdot d_{i-1}$. The optimization program of the generic scheme merely allows us to choose a good sequence from all the eligible l_i and d_i sequences satisfying (4).

We now discuss how to solve the optimization program. First, observe that it is a non-linear program, since the objective function (3) is non-linear. The distance sensitivity constraint (4) is also non-linear when g is a non-linear function. Second, notice that the number m is an unknown variable. Both characteristics make the solution to the program non-trivial. We use the following intuitive approach to solve the program: try a range of reasonable values of m; for each value of m, use standard non-linear programming techniques to solve the more constrained program. In the end, we choose the value of m that presents the best performance and the corresponding non-linear programming solution.

Choosing the best value for m is very important. Given a specific value of m, the program can be solved reasonably efficiently according to our experiments. In our experiments, all the results were obtained by the Matlab optimization toolbox [20] on a laptop with a 1.3GHz Intel PentiumM processor and 1Gb memory. Matlab solved all the programs successfully, some of which with m as large as 10000, in a matter of seconds.

In the following, we show the results for the generic scheme with a quadratic retrieval cost function $g(r) = r^2$. Table I lists the l_i sequence and the corresponding storage cost f (the objective function of the optimization program) for $m = 1, 2, 3, ..., 8$.

Table I. Solution of the optimization program with a quadratic retrieval cost function $g(r) = r^2$.

m	f	l_i
1	32046	1
2	286.9	1,21.9
3	118.3	1,5.5,71.4
4	100.1	1, 3.4, 15.0, 100
5	102.0	1, 3.1, 10.1, 34.0, 100
6	106.7	1, 3.0, 9.8,31.5, 86.0, 100
7	111.7	1, 3.0, 9.6, 29.8, 77.0, 100, 100
8	116.9	1, 3.0, 9.5, 28.6, 71.0, 100, 100, 100

As shown in Table I, when m increases from 1 to 8, the storage cost f first decreases monotonically and then increases. The optimal storage cost $f = 100.1$ is obtained when $m = 4$. Recall that the distance sensitivity of the range search scheme is

achieved by segmenting the distance between the data owner and query initiator (that is, the range $[1, D]$) into m sub-ranges. Table I shows that both over-segmentation and under-segmentation will result in high storage costs, and the optimal segmentation lies in between the two extremes. Furthermore, according to our experiments, this observation is true for not only the quadratic function, but also other retrieval cost functions as well. As a result, it suggests an alternative approach for solving the optimization program based on the binary search of m's value rather than the exhaustive search.

The generic scheme presented here not only works for non-linear retrieval functions, but also outperforms its baseline version with linear retrieval cost functions. Recall that a scheme with linear retrieval cost is one whose retrieval cost function is $g(r) = s \cdot r$. The comparison is illustrated in Table II, where the storage cost f and the l_i sequence are shown. The parameters are $D = 100$, and $s = 0.25, 0.5, 1, 2$.

Table II. The storage cost f and the l_i sequence of the baseline and generic schemes, $D = 100$.

	baseline		generic	
s	f	l_i	f	l_i
0.25	6729	1,2,4	6517.6	1,2.4,4.8
0.5	2614.5	1,2,4,8	2387.3	1,3,7.8
1	987.6	1,2,4,8,16	869.0	1,2.8,7.5,17.2
2	389.5	1,2,4,8,16,32	328.8	1,3.3,10.3,29.3

The following are two observations on the data in Table II. First, the storage cost of the generic scheme is indeed consistently lower than that of the baseline scheme.

In particular, when $s = 2$, the storage cost of the generic scheme is 18% less than the baseline version. Second, the baseline scheme tends to over-segment the distance range $[1, D]$. In particular, for $s = 1$ and $s = 2$, the baseline scheme divides the range into 4 and 5 sub-ranges, while the generic scheme achieves better performance by dividing it into only 3 and 4 sub-ranges.

Table III is a further illustration of the above results. It lists the storage cost f and the l_i sequence of the baseline and the generic schemes for $s = 2$ and $D = 25, 50, 100, 200$. We can see that the same conclusions hold.

Table III. The storage cost f and the l_i sequence of the baseline and generic schemes with $s = 2$.

	baseline		generic	
D	f	l_i	f	l_i
25	238.2	1,2,4,8	210.5	1,3,7.9
50	312.9	1,2,4,8,16	272.4	1,2.8,7.6,17.4
100	389.5	1,2,4,8,16,32	328.8	1,3.3,10.3,29.3
200	467	1,2,4,8,16,32,64	405.1	1,3.8,14.2,49.6

4. Load balancing

For ease of exposition, the design of the range search schemes presented in this section does not take load balancing into consideration. In this subsection, we describe a modified version of the range search schemes with more balanced storage load. We would like to point out that the discussion in this subsection also applies to the design of object locating schemes to be presented shortly in the next section.

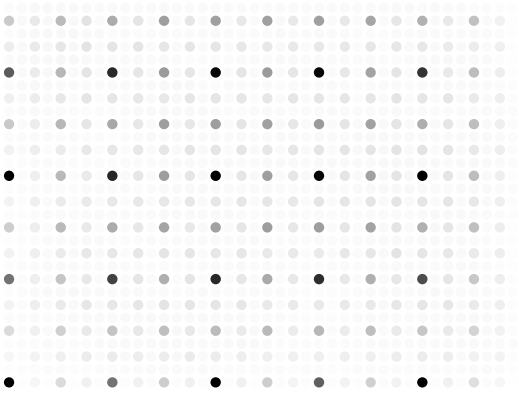


Fig. 6. Load distribution of the baseline range search scheme for a 40 by 30 grid with $s = 0.5$. The load of a node is indicated by its darkness.

Observe that, by the definition of the range search schemes, the storage load for each node is not balanced. In particular, in the baseline scheme, the 2^i-lattice nodes are chosen to be the storage servers, and according to the storage policy, a 2^i-lattice node will hold a copy of all the data items within distance $d_i + 2^i/\sqrt{2}$ from itself. Therefore, the storage load of the 2^i-lattice nodes tends to increase as i increases. This is illustrated in Figure 6, which depicts the storage load of all the nodes in a 40 by 30 grid network. There are 2048 data items in this example, and each data item is owned by a random node. The stretch factor s is set to be 0.5. The load of a node is depicted by its darkness. The darker a node is, the heavier it is loaded. As the figure shows, the nodes with the heaviest load are the 8-lattice nodes, followed by the 4-lattice nodes, and then 2-lattice nodes, and so on.

The design of our range search schemes made an implicit assumption that all the data items are of the same type. In practice, the data that users are interested in may naturally fall into a few different categories. For example, in a sensor network, a sensor may generate both temperature and humidity data. In what follows, we describe a load balanced version of the range search schemes by storing data in different

(a) (b)

Fig. 7. Load distribution of the load balanced range search scheme. The 2048 data items fall into (a) two, and (b) four categories.

categories at different lattice nodes, which only requires a small modification to the existing design. Observe that, the reason that the 2^i-lattice nodes are the performance "hotspots" is because all the data items share a common lattice system regardless of their types. And, the basic idea of the load balanced range search scheme is simply to use a different lattice system for each type of data. This can be achieved by a slight modification to the definition of d-lattice. In particular, we can define the set of d-lattice nodes for a type of data as the set of nodes with coordinate $(x + a \cdot d, y + b \cdot d)$, where a and b are integers ranging from $-\infty$ to $+\infty$, and (x, y) is the origin of the lattice system for that type of data. Each data type has its own lattice origin. It is straightforward to show that, the range search schemes with this modified notion of d-lattice are still correct and distance sensitive.

Figure 7 shows the load distribution of the load balanced range search schemes for the same network and set of data items in Figure 6. In Figure 7(a), the data items are categorized into two groups, and are stored in two corresponding lattice systems. In Figure 7(b), the data items are categorized into four groups. Clearly, the load

distribution in Figure 7 is considerably more balanced than Figure 6. The conclusion is also confirmed by many other experiments with different settings.

D. Object locating

In this section, we study distance sensitive object locating in distributed ad hoc networks, a problem that has a very similar structure as the range search problem discussed in the previous section. We first present the problem formulation in Subsection 1, and then present the design of the baseline and generic object locating scheme for grid networks in Subsection 2 and 3 with linear retrieval cost function and any arbitrary retrieval cost function respectively. In Subsection 4, we present a lower-bound result which shows that logarithmic storage cost is asymptotically optimal for a linear retrieval cost function. At last, in Subsection 5, we briefly discuss how to design object locating schemes for non-grid networks.

1. Problem statement

The range search problem discussed in the previous section studies regional data retrieval, while the object locating problem in this section studies the retrieval of individual data items. Specifically, we consider a set of objects in a network. Each object has a identifier and a value, and is owned by a node, called its owner. A *reference* to an object is an identifier-address pair, which indicates the address of the owner of the object with the identifier. A node that stores a reference to an object is called the reference server of the object. Note that, in the range search problem, we store the data items directly at the storage servers, while in object locating, we store object references at the reference servers. When a node tries to find an object, it initiates a query with the identifier of the object, and the query is forwarded along

a pre-designed path, called the *retrieval path* (of the query initiator), until a node which holds a reference to the object is reached. The *storage cost* of an object is the total number of its reference servers, and the storage cost of an object locating scheme is the maximum storage cost over all objects. When a node is searching for an object, the *retrieval cost* is the length of the retrieval path segment between the node and the first node on its retrieval path that has a reference to the object.

An object locating scheme defines both a storage and a retrieval policy. The storage policy specifies where to place references to each object in the network, while the retrieval policy specifies the retrieval path for every query initiator and every object. An object locating scheme is *distance sensitive* with the retrieval function g if for any node and any object, the retrieval cost for the node to find the object is at most $g(r)$, where r is the distance between the query initiator and the object owner. Similar to the range search problem, there is an intrinsic tradeoff between the storage and retrieval cost of an object locating scheme. In general, the higher the storage cost, the lower the retrieval cost.

2. The baseline scheme

The baseline version of the object locating scheme is designed to achieve linear retrieval cost. Specifically, the retrieval cost function is $g(r) = s \cdot r$, where s, the stretch factor, is an arbitrary positive number.

The baseline object locating scheme is defined as follows:

- *Storage policy:* a node stores a reference to any object it owns at all the 2^i-lattice nodes within distance $d_i + 2^i/\sqrt{2}$ of itself, $i = 0, 1, 2, \cdots, m = \lceil \log(s \cdot D/\sqrt{2}+1) \rceil$, where

$$d_i = \sqrt{2} \cdot (2^i - 1)/s. \tag{7}$$

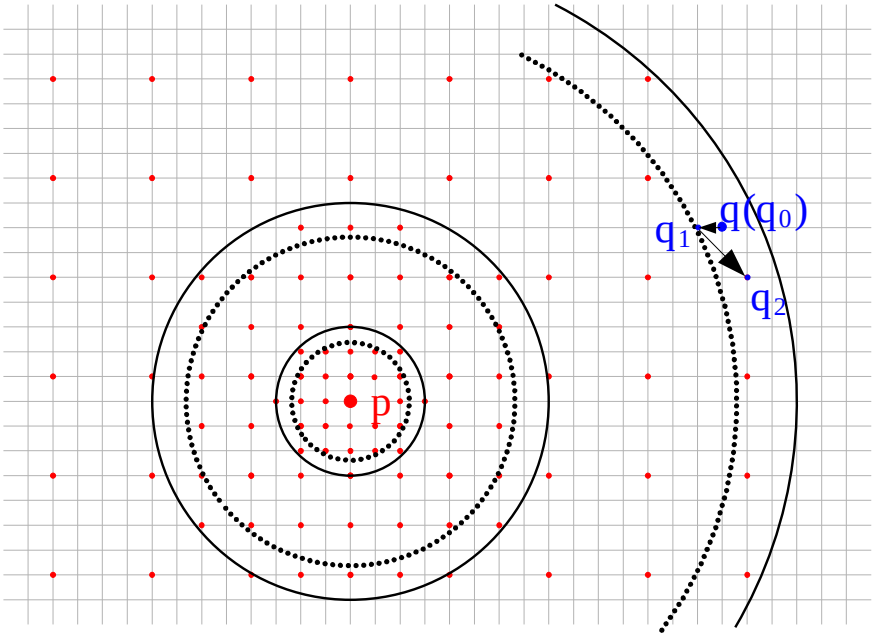


Fig. 8. The baseline object locating scheme. p is the object owner and q is the query initiator. q_1 and q_2 are the 2- and 4- neighbors of the node q. q finds p at q_2.

- *Retrieval policy:* a node q retrieves a target object by searching along the path $q_0 \rightarrow q_1 \rightarrow q_2 \cdots$, where q_i is the 2^i-neighbor of q, regardless of the identifier of the object.

The baseline scheme is illustrated in Figure 8. The node p is the owner of the target object that q is looking for. The reference servers of the node p are shown in the figure with black dots. q_1 and q_2 are the 2- and 4- neighbors of the node q. q_0 and q are the same node. In this figure, q_2 is a reference server for p, and at the same time, it is on the retrieval path of q.

Now, we analyze the storage policy. Consider an arbitrary object. There are m layers of reference servers for the object. By the definition of m, $m = O(\log D)$. In each layer, the number of reference servers is upperbounded by $\pi \cdot (d_i + 2^i/\sqrt{2})/(2^i)^2$.

By the definition of d_i, it is straightforward to see to that $d_i = O(2^i)$. Therefore, the number of reference servers in each layer is $O(1)$, and the total number of reference servers, i.e., the storage cost of the object, is $O(\log D)$.

Now, we analyze the retrieval policy. Let p be the owner of the target object that q is looking for. First, we show that, if $|pq| \leq d_j$ for some integer $j \geq 0$, then q_j, the 2^j-neighbor of q, is a reference server for the object. By the definition of 2^j-neighbor, we have $|qq_j| \leq 2^j/\sqrt{2}$. Since $|pq| \leq d_j$, $|pq_j| \leq |pq| + |qq_j| \leq d_j + 2^j/\sqrt{2}$. According to the storage policy, q_j is one of p's reference servers. Since $d_m \geq D$ and the distance between any two nodes is at most D, it follows that the 2^m-neighbor of any node holds a reference to any object in the network and the retrieval process in the worst case ends at the 2^m-neighbor of the query initiator. Now, we show that the retrieval policy is distance sensitive. More specifically, we show that the cost for q to find the object owned by p is at most $s \cdot |pq|$. Let j be the smallest integer such that $|pq| \leq d_j$. We have just shown that q_j is a reference server for any objects owned by p. To prove that the retrieval policy is distance sensitive, we just need to show that the length of the path $qq_1q_2 \cdots q_j$, is bounded by $s \cdot |pq|$. Specifically, this length is,

$$\sum_{i=1}^{j} |q_{i-1}q_i| \leq \sum_{i=1}^{j} (|qq_{i-1}| + |qq_i|) \leq \sum_{i=1}^{j} 2|qq_i|,$$

by the definition of q_i, $|qq_i| \leq 2^i/\sqrt{2}$, therefore, the length of this path is at most

$$\sum_{i=1}^{j} 2 \cdot 2^i/\sqrt{2} = \sqrt{2} \cdot (2^{j+1} - 1) = s \cdot d_{j-1},$$

since j is the smallest integer such that $|pq| \leq d_j$, it must be true that $d_{j-1} \leq |pq|$, therefore, the length of this path is at most $s \cdot |pq|$, which implies that the retrieval cost is at most $s \cdot |pq|$.

Theorem 3. *The baseline scheme has $O(\log D)$ storage cost and $O(r)$ retrieval cost.*

3. The generic scheme

In this section, we generalize the baseline version presented in the previous subsection, and present the generic object locating scheme with an arbitrary retrieval cost function.

The generic scheme is defined as follows:

- *Storage policy:* a node stores a reference to any object it owns at all the l_i-lattice nodes within distance $d_i + l_i/\sqrt{2}$ from itself, $i = 0, 1, 2, \cdots, m$, where m is the smallest integer such that $d_m \geq D$.
- *Retrieval policy:* A node q retrieves a target object by searching along the path $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \cdots$, where q_i denote the l_i-neighbor of q.

The sequences l_i and d_i are solutions of the following optimization program:

$$\min \quad \sum_{i=0}^{m} \pi(d_i + l_i/\sqrt{2})^2/l_i^2 \tag{8}$$

$$s.t. \quad \sqrt{2}\sum_{j=1}^{i} l_j \leq g(d_{i-1}) \quad \forall 1 \leq i \leq m \tag{9}$$

$$d_i \geq 0, l_i \geq 0 \qquad \forall 0 \leq i \leq m \tag{10}$$

$$d_0 = 1, d_m \geq D \tag{11}$$

In the optimization program, the objective function, $\sum_{i=1}^{m} \pi(d_i + l_i/\sqrt{2})^2/l_i^2$, is an upperbound of the storage cost of an object. The inequality (10) is the non-negativity constraint.

Now we analyze the generic scheme. Let p be the owner of the object that q is looking for. By the same argument presented in the previous subsection, we know that q_j is a reference server of any objects owned by p, where j is the smallest integer such that $|pq| \leq d_j$. The retrieval cost is at most the length of the path $qq_1q_2 \cdots q_j$,

which is $\sum_{i=1}^{j} |q_{i-1}q_i| \leq 2 \cdot \sum_{i=1}^{j} |qq_i|$. Since q_i is the l_i-neighbor of q, we know that $|qq_i| \leq l_i/\sqrt{2}$. Therefore, the retrieval cost is at most $2 \cdot \sum_{i=1}^{j} l_i/\sqrt{2} = \sqrt{2} \cdot \sum_{i=1}^{j} l_i$, which is at most $g(d_{j-1})$ by (9). Since j is the smallest integer such that $|pq| \leq d_j$, it must be true that $d_{j-1} \leq r$. Therefore, the retrieval cost is at most $g(r)$, which concludes the following theorem,

Theorem 4. *The generic object locating scheme is distance sensitive with a retrieval function g.*

4. Lowerbound result

In this subsection, we analyze the optimality of the performance of the distance sensitive object locating scheme.

Linear retrieval cost is a very popular distance sensitive constraint, and has been investigated in a large number of previous studies [21, 13]. The design of the object locating schemes presented in this chapter has a storage cost that is logarithmic in the diameter of the network. A natural question is, is this the optimal storage cost to achieve the linear retrieval constraint? We prove in this section that it is true. We show that for a wide family of networks called the growth lowerbounded networks, the logarithmic storage cost is a lowerbound for any object locating scheme that achieves linear retrieval cost. Since growth lowerbounded networks includes the grid networks as a special case, the baseline scheme achieves the asymptotically optimal performance.

Growth bounded networks characterize a large family of networks, and have been widely studied [16]. The notion of *bounded growth* is based on the concept of *ball packing*. A *ball* $B_r(v)$ with center node v and radius r is defined as the set of nodes within distance r from v. Here, the distance between two nodes is the routing

cost between the two nodes. Let $\mathcal{B}$ be a set of balls each with radius r, and let $U \subseteq V$ be a subset of nodes in the network. $\mathcal{B}$ is called a *r-packing* of U if $\cup_{B \subseteq \mathcal{B}} B \subseteq U$ and $\forall B, B' \in \mathcal{B}$, $B \cap B' = \Phi$. That is, $\mathcal{B}$ is a set of mutually non-intersecting balls each with radius r. A network is *growth lowerbounded* or simply *growth bounded* with growth rate α if for any $v \in V$ and any r, r' such that $r' \leq r$, there exists a r'-packing of at least $(r/r')^\alpha$ balls for $B_r(v)$.

Consider an arbitrary distance sensitive object locating scheme with a linear retrieval cost function. Without loss of generality, we assume that the minimum distance between any two nodes is one. Let s be the stretch factor in the linear retrieval cost function $g(r)$, that is, $g(r) = s \cdot r$. An important quantity in the following analysis is the *zooming factor*, which is defined as:

$$z = z(s, \alpha) = (1.5s)^{\frac{\alpha}{\alpha - 1}}.$$

Before presenting the following lemma (Lemma 5), we introduce a few more terms. The distance $|vP|$ between a node v and a path P is the minimum distance between v and any node in P. We say that a path *crosses* a ball if the intersection of the (set of nodes on that) path and the (set of nodes in the) ball is non-empty. Intuitively, the following lemma says that for any ball and any (retrieval) path starting from its center, there is a big enough (sub-)ball contained in the ball such that the path does not cross.

Lemma 5. *Let v be an arbitrary node in a growth bounded network. Let $P = (v, w_1, w_2, ...)$ be a path whose length $|P|$ is at most $s \cdot r$. Then, there must exist a node v' such that $B_{r'}(v') \subseteq B_r(v)$ and $|v'P| > s \cdot r'$, where $r' = r/z$.*

Proof. For simplicity, in the following analysis, we let B' denote $B_{r'}(v')$, and let B denote $B_r(v)$. See Figure 9 for an illustration.

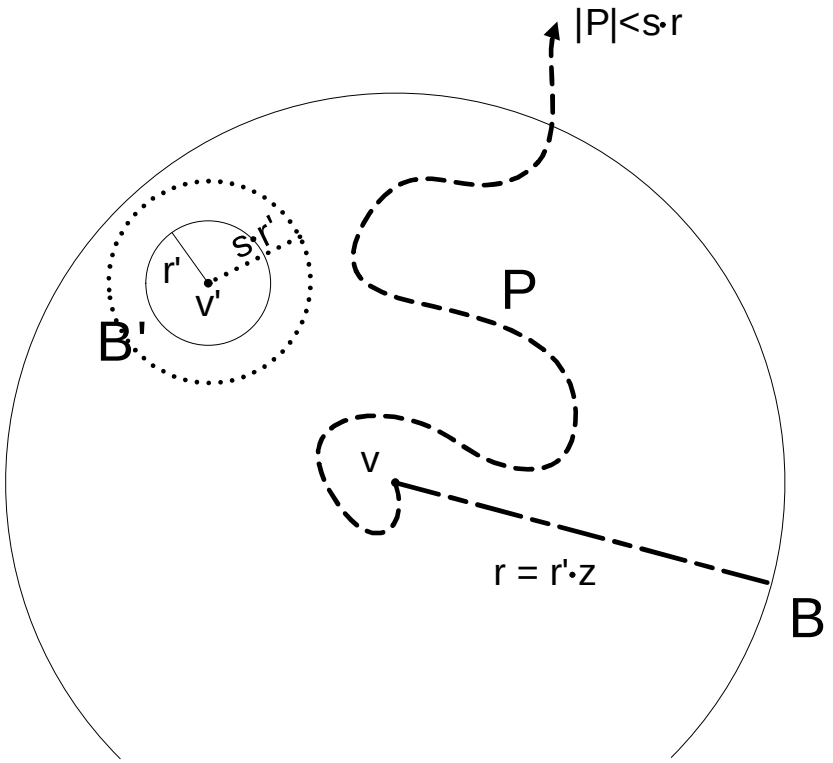


Fig. 9. Lemma 5. It shows that for any ball and any (retrieval) path starting from its center, there is a big enough ball contained in it such that the path does not cross.

By the basic property of the bounded growth of the network, there is a $1.5sr'$-ball packing of cardinality

$$(\frac{r}{1.5sr'})^{\alpha} = (\frac{z}{1.5s})^{\alpha} = (\frac{1.5s^{\alpha/(\alpha-1)}}{1.5s})^{\alpha} = (1.5s)^{\frac{\alpha}{\alpha-1}} = z$$

for any ball of radius r. Therefore, there is such a packing for $B = B_r(v)$. Shrink the radius of each ball in this packing from $1.5sr'$ to sr', and we obtain an sr'-ball packing $\mathcal{B}$ for B of cardinality z. Clearly, the distance between any two balls in this packing is at least sr', as shown in Figure 10.

Consider the number of balls in $\mathcal{B}$ that P can cross. Since $|P| \leq s \cdot r$ and

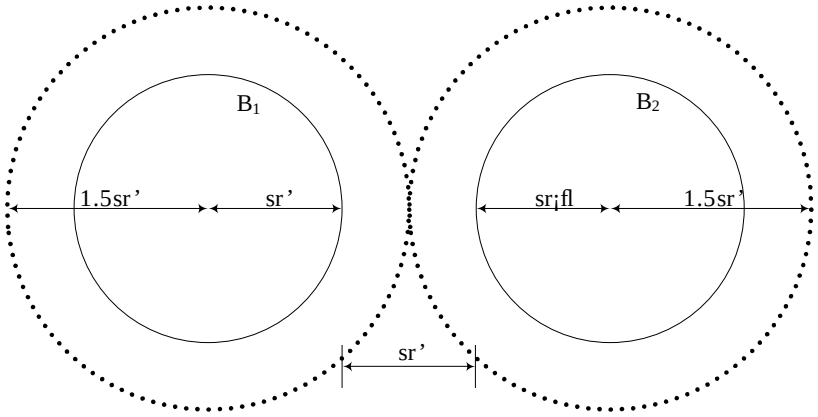


Fig. 10. Two balls B_1 and B_2 in an sr'-ball packing obtained from a $1.5sr'$-ball packing, where the distance between B_1 and B_2 is at least sr'.

the minimum inter-ball distance of $\mathcal{B}$ is at least $s \cdot r'$, P needs to "spend" at least $s \cdot r'$ in order to go from one ball to anther, and therefore P can cross at most $s \cdot r/(s \cdot r') = r/r' = z$ balls in $\mathcal{B}$. Since there are at least z balls in $\mathcal{B}$, there is at least one ball that P cannot cross. Let v' be the center of this ball. Obviously $B_{s \cdot r'}(v') \subseteq B$ because $B_{s \cdot r'}(v')$ is a member of the ball packing $\mathcal{B}$. Also, $B' = B_{r'}(v') \subseteq B$ too because $B_{r'}(v') \subseteq B_{sr'}(v')$. By our choice of v', P does not cross $B_{sr'}(v')$, which implies that $|v'P| > sr'$. Therefore, v' is the node we have been looking for. □

The above proof also explains how the expression of z is chosen. It ensures that the number of balls in $\mathcal{B}$ is more than the number of balls P can cross. In other words, z is specifically chosen so that Lemma 5 will hold. In fact, z can be clearly set to any constant less than $(1.5s)^{\alpha/(\alpha-1)}$.

In the proof of the following theorem, Lemma 5 is applied repeatedly as a basic building block of the argument.

Theorem 6. *Let B be an arbitrary ball of radius R in a growth bounded network. For any object locating scheme with the linear lookup cost function $g(r) = s \cdot r$, there*

must exist a node $p \in B$ such that the storage cost of any object it owns is $\Omega(\log R)$.

Proof. The basic idea of this proof is to repeatedly apply Lemma 5, and construct a sequence of $\Omega(\log R)$ nested balls $B_1, B_2, ...$ with exponentially decreasing radius, a sequence of nodes $q_1, q_2, \ldots$, and a sequence of mutually non-intersecting lookup paths $P_1, P_2, \ldots$ with $q_1, q_2, \ldots$ as their starting ends. The construction ensures that the node sitting at the center of the innermost ball must have one reference server on each of the $\Omega(\log R)$ retrieval paths. See Figure 11 for an illustration.

First, let q_1 be the center of B. Let $B_1 = B_{r_1}(q_1)$, where r_1 is the radius of B_1. Let P_1 be the maximum segment of q_1's retrieval path for the target object such that $|P_1| \leq s \cdot r_1$. By Lemma 5, there is a node q_2 such that ball $B_2 = B_{r_2}(q_2) \subseteq B_1$ and $|q_2P_1| > s \cdot r_2$, where $r_2 = r_1/z$. Let P_2 be the maximum segment of q_2's retrieval path such that $|P_2| \leq s \cdot r_2$. Notice that $|q_2P_1| > s \cdot r_2$ and $|P_2| < s \cdot r_2$, that is, the distance from node q_2 to path P_1 is at least $s \cdot r_2$, while the length of P_2 starting from q_2 is less that $s \cdot r_2$, therefore $P_1 \cap P_2 = \Phi$, that is, P_1 and P_2 are mutually disjoint. Again by Lemma 5, there is a node q_3 such that ball $B_3 = B_{r_3}(q_3) \subseteq B_2$ and $|q_3P_2| > s \cdot r_3$, where $r_3 = r_2/z$. By repeating this process in $m = \log R / \log z$ steps, we will have a sequence of nodes $q_1, q_2, ..., q_m$, a sequence of balls $B_1, B_2, ..., B_m$ and a sequence of paths $P_1, P_2, ..., P_m$. By the above construction, $B_m \subseteq B_{m-1} \subseteq ... \subseteq B_2 \subseteq B_1$, and $P_1, P_2, ..., P_m$ are mutually disjoint.

We can now show that q_m is the node we are searching for. To verify this, let $p = q_m$ be the owner of the target object. It is straightforward to see that $\forall 1 \leq i \leq m$, $p \in B_i$ because $p \in B_m$ and $B_m \subseteq B_i$. The fact that $p \in B_i$ implies that $|pq_i| \leq r_i$. Since the retrieval stretch factor is s, and P_i is the maximum segment of the q_i's retrieval path satisfying $|P_i| \leq s \cdot r_i$, p must have at least one reference server on P_i ($\forall 1 \leq i < m$) in order for q_i to locate it with a cost no more than

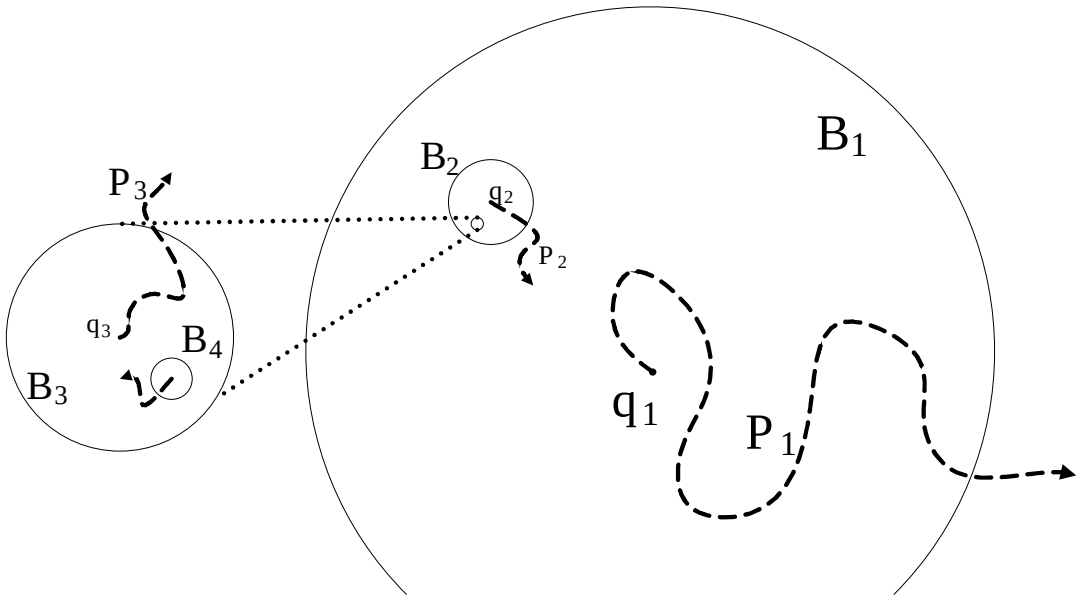


Fig. 11. The proof of theorem 6. It shows the construction of a sequence of non-intersecting paths P_1, P_2, P_3..., and a sequence of balls $B_1 \supseteq B_2 \supseteq B_3$... with exponentially decreasing radii.

$s \cdot |pq_i|$. Since $P_1, P_2, \ldots, P_{m-1}$ are mutually non-intersecting, the total number of p's reference servers in B is at least $m - 1 = \log R / \log z - 1 = \Theta(\log R)$. □

Theorem 6 shows that for a growth-bounded network of diameter D, for any distance sensitive object locating scheme with linear retrieval cost, its storage cost must be at least $\log D$ times a constant. Therefore, the following corollary directly follows.

Corollary 7. *The baseline object locating scheme achieves asymptotically optimal storage cost.*

We would like to note that there are a few object locating methods in literature that also achieve $O(\log D)$ storage cost. An example is the location service scheme LLS presented in [13]. Our result also proves the optimality of those schemes.

Due to the similarity between the range search and object locating problem, one might hope that the same proof technique can be applied to show the optimality of the range search schemes presented in the previous section. Unfortunately, we were unable to do that directly, and the optimality of the range search schemes remains unknown to us.

5. Beyond grid networks

The design of the range search and object locating schemes so far is based on a fairly restricted network model, the grid network. In practice, however, the network under study usually has more complicated topology, and the design of the object locating schemes does not directly apply. In this subsection, we describe how to design object locating schemes for growth upper bounded networks. We would like to point out that, the techniques presented in this subsection also directly apply to the design of range search schemes.

First, we use a simple example to explain the limitations of the object locating schemes that we have developed. Figure 12(a) shows a grid where the 4-lattice nodes are depicted as black dots. The node q_4, as the 4-neighbor of the node q, is on q's retrieval path. For the gird in Figure 12(a), q can easily calculate the position of q_4 according to the position of itself, and $|qq_4| \leq 2^4/\sqrt{2}$. These two facts are necessary conditions for the retrieval policy to be correct and distance sensitive. Figure 12(b) shows the same grid with two holes in the middle of the network. The node q_4, which was present in Figure 12(a), happens to be in the hole area and does not exist in Figure 12(b). In addition, the distance between q and its nearest 4-lattice node in Figure 12(b) is more than $2^4/\sqrt{2}$, therefore, even if q is aware of the existence of the holes, and picks the q's nearest 4-lattice node as its 4-neighbor, the distance sensitivity property of the retrieval policy would still be violated.

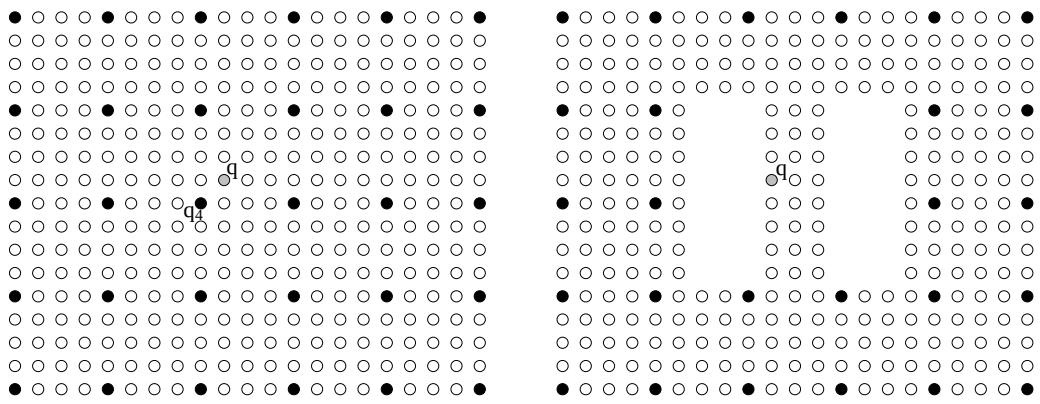


Fig. 12. A grid network with two holes in the middle. The 4-lattice nodes are depicted as black dots.

The basic principle of the grid based distance sensitive object locating schemes can be generalized and used for designing object locating schemes for a class of networks called growth upper bounded networks. The notion of upper bounded growth is based on the notion of ball covering. Let $\mathcal{B}$ be a set of balls each with radius r, and let $U \subseteq V$ be a subset of nodes in the network. $\mathcal{B}$ is called a *r-covering* of U if $U \subseteq \cup_{B \subseteq \mathcal{B}} B$. A network is *growth upper-bounded* with *growth rate* β if for any $v \in V$ and any r, r' such that $r' \leq r$, there exists a r'-covering of at most $(r/r')^\beta$ balls for $B_r(v)$. Apparently, the notions of upper and lower bounded growth are a pair of dual concepts.

Recall that, the basic principle of the grid based schemes is to use d-lattice nodes as places for information storage and retrieval. Correspondingly, the basic principle for object locating schemes for growth upper bounded networks is to use d-net nodes as rendezvous places, where the notion of d-net is a natural generalization of the concept of d-lattice. Let V denote the set of nodes in the network. A node set $U \subseteq V$ forms a d-net of V if the distance between any pair of nodes in U is at least d, and for

any node $v \in V - U$, there is at least one node $u \in U$ such that the distance between u and v is at most d. It is well-known that, a d-net can be obtained by the following simple greedy algorithm:

1. Initially, set V to be the set of all nodes and U to be the empty set,
2. pick an arbitrary node v in $V - U$, put it in U, and remove all the nodes whose distance to v is less than d,
3. repeat the step 2 until V is empty.

When the algorithm terminates, the set U is a d-net. An important characteristics of a growth upper bounded network is, the number of d-net nodes in any ball of radius r is at most $2^\beta \cdot (r/d)^\beta$. This can be shown as follows. Consider an arbitrary ball B with radius r. By the definition of growth upper bounded network, there exists a $d/2$-ball covering of cardinality at most $(2r/d)^\beta = 2^\beta (r/d)^\beta$. Notice that, since the radius of each ball in the covering is $d/2$, and the distance between any pair of d-net node is at least d, there must be at most one d-net node in each ball in the covering. This implies that, the number of d-net nodes in B is at most $2^\beta \cdot (r/d)^\beta$. It follows directly from this fact that the number of d-net nodes in any ball of radius $O(d)$ is $O(1)$, where the growth rate β is considered as a constant.

By the above characteristics of growth upper bounded networks, the following object locating scheme achieves logarithmic storage and linear retrieval cost:

- *Storage policy:* a node stores a reference to each object that it owns at all the 2^i-net nodes within distance $d_i = O(2^i)$ from itself, $i = 0, 1, 2, \cdots, m$, where m is the smallest integer such that $d_m \geq D$.
- *Retrieval policy:* a node q retrieves the target object by searching along the path $q_0 \rightarrow q_1 \rightarrow q_2 \cdots$, where q_i is the nearest 2^i-net node of q.

The above scheme for growth upper bounded network looks almost identical as the baseline scheme designed for gird networks, and it is straightforward to show that the above object locating schemes is correct and distance sensitive. Despite the obvious similarity, the above schemes is actually a little more involved than the grid based scheme in the sense that to store and retrieve information according to this policy, the nodes in the network must be aware of a sequence of pre-constructed 2^i-nets. More specifically, to store object references, an object owner must "remember" the addresses of all the 2^i-net nodes within distance d_i from itself. Note that, in grid networks, the object owner can simply "calculate" the locations of all the 2^i-lattice nodes within distance d_i. Similarly, to retrieve an object, a node needs to remember the addresses of the nodes on its retrieval path, while in grid networks, the locations of the nodes on the retrieval path can also be "calculated". In summary, object locating schemes for growth bounded networks need to maintain a certain amount of information of the topology of the network, which is not necessary for grid networks. This additional overhead comes at no surprise. After all, grid is a special kind of growth bounded network, and the topology of an arbitrary growth bounded network does not have the regularity that a grid has.

E. Related work

A distinct characteristic of the information storage and retrieval schemes is that they are designed for geometric networks, which can be embedded in 2- or 3-dimensional Euclidean space. Such data storage and retrieval schemes for geometric networks have been an active topic of study, especially in the context of sensor network research. A well known approach is the Geographic Hash Table (GHT) [22], where a hash function is used to determine the rendezvous locations to store data. It uses the geographic

routing algorithm GPSR [6] to transmit messages. GPSR guarantees that messages can be routed to the node geographically nearest the rendezvous location. The same hash function is used by both information producers and consumers, so that the consumers know where to find the data associated with a key. The approach assumes that sensors know their physical locations. GHT is not a distance sensitive scheme. For a grid network with n nodes and a geographic hash function with uniform distribution, the average lookup cost for GHT is $O(\sqrt{n})$. Distance sensitive information brokerage has been studied in [21], where the authors proposed a novel double ruling based information brokerage scheme which we refer to as DR. The key idea of the DR scheme is to store the data replicas on continuous curves instead of one or multiple isolated nodes. The curve along which a producer replicates its data is designed in a way that guarantees it to intersect the lookup path of any consumer. For a network of n nodes, a producer needs to store its data in $O(\sqrt{n})$ rendezvous nodes on the curve. The scheme achieves linear lookup cost. In addition, DR supports structured aggregate queries, meaning that a consumer following a particular curve can retrieve the data from all producers in the network. The authors argue that the flexibility of their retrieval mechanisms gives better routing and data robustness. Additional information storage schemes based on landmarks, etc., have been proposed in [23, 24].

Probably the most related research to ours is distributed object locating and routing (DoLR) schemes. DoLR is usually studied in the context of efficient file sharing in large scale distributed networks, especially overlay networks. The main difference between DoLR and the distributed object locating problem discussed in this chapter is that, DoLR schemes include an extra phase to construct an overlay network on top of the original network, and an efficient routing scheme for the overlay network. In DoLR, objects are mapped to overlay network nodes, and then the problem of finding an object is reduced to the problem of finding the overlay node(s)

where the object is mapped to. DoLR comes in two flavors: distance sensitive and insensitive.

Research in DoLR has received significant attention during the last decade. The two early popular schemes are Chord [25, 26] and CAN [27]. In Chord, network nodes are organized as an identifier ring. In a network with n nodes, each node maintains information about $O(\log n)$ other nodes, and each lookup request is resolved in $O(\log n)$ steps. The other original DoLR proposal, CAN, is built around a virtual d-dimensional Cartesian coordinate space on a multi-torus. The entire coordinate space is dynamically partitioned among all the n nodes in the network such that every node possesses its individual, distinct zone within the overall space. Each node maintains information about $O(d)$ other nodes, and each lookup request is resolved in $O(dn^{1/d})$ steps. Following Chord and CAN, a large number of DoLR schemes were developed, noticeably Koorde [28], Kademlia [29], Viceroy [30], Symphony [31], and etc. All these schemes focus on system scalability and maintainability, while disregarding data locality. In the seminal work by Plaxton et al.[16], a randomized distance sensitive DoLR solution was proposed for growth bounded networks, which was called the PRR scheme. The schemes Tapestry [18] and Pastry [19] inherited the basic ideas of PRR, and focused more on the self organization of overlay networks under frequent node arrivals, departures and failures. PRR, Tapestry and Pastry all have linear lookup cost in expectation. LAND [17] is a DoLR scheme for growth bounded networks. It achieves a deterministic $1 + \epsilon$ stretch lookup cost in the worst case.

Another closely related topic is target tracking (or location service) in mobile networks, where a node uses the data stored in location servers to learn the locations of other nodes. The study of location service was pioneered by Awerbuch and Peleg [32][33]. In their work, the network was modeled by a general graph, and the

approach was based on hierarchical regional directories. LLS [13] adopted the ideas in [32][33] on tracking, and obtained a distance sensitive scheme with linear lookup ratios. It replicates data in $\log n$ nodes for grid-like networks. STALK [15] and MLS [14] achieved a similar tracking performance. They also addressed the issues of fault tolerance and robust performance. In particular, MLS has studied the maximum speed at which the nodes can move in order to guarantee locality-sensitive lookup. The above schemes all have linear lookup costs. GLS (grid location service) is another interesting scheme [34]. It is based on the hierarchical decomposition of the network using a quad-tree structure, and location servers are selected from the network components in the quad-tree. Although its lookup cost is not locality-sensitive in the rigorous sense, it does enable a consumer to find the location information of a closer node from a location server in a smaller region. Therefore it achieves a performance that is nearly locality sensitive.

Finally, distance sensitive information retrieval is also related to compact routing. Compact routing comes in two flavors, labeled routing and name independent routing. The objective of both of the two problems is to find a short (not necessarily the shortest) path between node pairs while minimizing the size of routing tables. The main difference between the two variants is that in labeled routing, the names (or addresses) of the nodes are assigned by the system designer according to the network topology, while in name independent routing, the names are chosen by the users. The range search and object locating problems are more related to the name independent routing problem. Name independent compact routing has been extensively studied for a variety of network topologies, such as trees [35], Euclidean metrics [36], growth bounded networks [37], networks with low doubling dimension [38, 39, 40, 41], and general graphs [42, 43].

F. Summary

This chapter studies distance sensitive information services in distributed ad hoc networks. In particular, we consider two closely related problems, range search and object locating. The first main result of this chapter is a generic framework for designing distance sensitive services for grid networks. The framework allows the system designer to specify an arbitrary retrieval cost function, and outputs a scheme with optimized storage cost. To our best knowledge, this is the first time that general retrieval cost functions are studied for distance sensitive services. This framework is applicable to both the range search and object locating problems. Although we did not get into further discussion, the technique also be used in the problem of mobile target tracking to optimize the initialization cost as well as the update cost. The other main contribution of this chapter is a lower bound result for the object locating problem, which shows that an logarithmic storage cost is asymptotically optimal for schemes with linear retrieval cost in growth lower bounded networks. This result proves the asymptotic optimality of many existing work in the area of ad hoc networks as well as P2P networks research community. In addition, we also address other issues such as load balancing and non-grid network topology in the design of distance sensitive information services.

There are many interesting problems that are left unsolved by our study. First, despite of the similarity of the two problems studied in this chapter in terms of the problem structure and our solution framework, we only obtained a optimality result for the object locating problem, and have not been to obtain the same result for the range search problem even though we believe the result also holds. Second, the optimality result that we have obtained is still in a weaker form in the following sense. It only proves that to guarantee linear retrieval, at least one node in a growth lower

bounded network needs to have a logarithmic number of reference/storage servers. We suspect that a much stronger result also hold: to guarantee linear retrieval, most of the nodes in the network need to have a logarithmic number reference/storage servers. Apparently, the vague and intuitive notion of "most nodes" in the above statement need to be quantified. Third, we only obtained the optimality result when the retrieval cost function is a linear one. An intriguing question is, is the generic scheme that we have developed in this chapter also optimal in some sense for other retrieval cost functions? Note that, we identified that the lower bounded growth of a network is the necessary condition for the optimality result to hold in the linear retrieval case, is it also a necessary condition in the non-linear retrieval case? Fourth, we assumed that the network under study is static. The topology of practical networks, especially cellphone network, sensor network, and Internet, is dynamic. Nodes in these networks come and go on a regular basis, and it is of great interest to incorporate such network dynamics into the design of our distance sensitive information services. Fifth, the baseline scheme for both the two problems has very nice closed form storage and retrieval cost functions, the logarithmic and linear functions. In the generic scheme, however, the storage cost function does not necessarily take an closed form, which raises the following question. Is there any other closed form storage and retrieval cost function pairs? In summary, the general problem of distributed distance sensitive information retrieval is still far from completely solved, and our study is a step towards the goal.

CHAPTER III

END-TO-END INFORMATION TRANSMISSION

A. Introduction

A large number of ad hoc networks are wireless networks. An important characteristic of wireless networks is that wireless transmissions may conflict with each other due to various reasons such as radio interference, and therefore may not occur at the same time. As a result, wireless transmissions usually need to be carefully scheduled, according to certain schedulability constraints, to avoid such conflicts. This need of making conflict free schedules for transmissions adds considerable complications to many tasks during the network operation. In this chapter, we study the impact of various wireless schedulability constraints to two fundamental networking problems, end-to-end on-demand bandwidth allocation, and end-to-end throughput estimation.

The goal of an end-to-end on-demand bandwidth allocation scheme is to allocate available network resource to satisfy a user's end-to-end connection request with a specific bandwidth requirement, while optimizing the bandwidth utilization such that more future requests can be satisfied. In other words, we have two simultaneous objectives, guaranteeing quality of service (QoS) to the end users, and minimizing system resource consumption.

The end-to-end throughput estimation problem simply asks what is the maximum amount of data that can be successfully delivered between a pair of source and sink nodes in a given period of time. End-to-end throughput is a fundamental network parameter that can be used to identify potential bottleneck, coordinate network traffic, plan and evaluate network design, etc. It is generally difficult to compute the exact end-to-end throughput under various schedulability constraints, and also, it is

often satisfactory to have a reasonably small range within which the thoughput falls in. Because of this, in this chapter, we will only focus on efficient methods that give a reasonably good upperbound of the thoughput. In particular, we show that, under a widely adopted interference model, a practically tight upperbound can be obtained by identifying a small set of carefully chosen cliques.

Before getting into the detailed discussion of these two end-to-end information transmission problems, we first introduce some basic concepts and terminologies.

B. Basic concepts and terminologies

We consider an ad hoc network formed by a group V of homogeneous nodes. Each node is equipped with a wireless transceiver. All the other nodes that a node can directly communicate with (transmit to or receive from) successfully via its transceiver are called its neighbors. There is a directed link from a node to each of its neighbors. The set of all links is denoted by E. Let $e = uv \in E$ be a directed link, u and v are respectively called the sender and receiver of e. u is called v's predecessor and v is u's successor. Given a node v, $E_{in}(v)$ and $E_{out}(v)$ respectively denote the set of inbound and outbound links incident on v. Communication between two non-neighboring nodes is forwarded by one or more intermediate nodes. For two nodes $s, d \in V$, an $s \rightarrow d$ path p is a sequence of adjacent links connecting s and d. If $e = uv$ is a link of path p, we say $u \in p$, $v \in p$, and $e \in p$.

A notion of great importance in the discussion of this chapter is that of schedulability of transmissions. A *transmission* is a one hop communication that occurs at a particular link. The schedulability constraint characterizes the condition under which wireless transmissions can successfully occur at the same time. Formally, the *schedulability constraint* is a binary relation $\nparallel$ on the link set E. If $e_1 \nparallel e_2$, we say link

e_1 and e_2 *conflict with* each other, and e_1 and e_2 cannot be used for data transmission at the same time. The schedulability relation essentially defines a dependency among links, which could be caused by various reasons such as the ability of radio transceivers and radio interference. The schedulability constraint is the main complication factor for the problems to be discussed in this chapter. In particular, we consider the impact of the single transceiver constraint to the problem of end-to-end on-demand bandwidth allocation and the impact of radio interference to the problem of end-to-end throughput estimation.

C. Bandwidth allocation

In this Section, we first present a combinatorial optimization formulation of the on-demand end-to-end bandwidth allocation problem with the single transceiver schedulability constraint in Subsection 1. We analyze the complexity of this problem and show that it is NP-hard in Subsection 2. A 2-approximation algorithm MCRS (minimum consumption routing and scheduling) is presented in Subsection 3. Experiments show that MCRS significantly outperforms existing minimum hop based routing schemes in Subsection 4.

1. Problem statement

We first introduce the relevant notations and formulate the problem of end-to-end bandwidth allocation with a particular schedulability constraint, called the *single transceiver constraint*.

The single transceiver constraint says that a set of transmissions can successfully occur at the same time only if no two transmissions share a common sender or receiver. This constraint is caused by the fact that each node is equipped with only one radio

transceiver, and thus a node cannot transmit to two different receivers or receive from two different senders at the same time. The impact of the single transceiver constraint to routing and scheduling is the focus of our study. The effect of other type of schedulability constraint such as those caused by radio interference is not considered in this section. This simplification is based on the fact that interference can be dramatically suppressed with technologies such as directional antenna [44] and adaptive transmission power control [45]. Furthermore, interference can also be eliminated by a well-studied preprocessing step called channel assignment [46, 47, 48], which assigns interfering links to orthogonal channels that are mutually isolated.

We assume that all the wireless links between neighboring nodes have the same capacity. The network works on the basis of TDMA (time division multiple access), where time is divided into equal length *frames* and each frame into equal length *slots*. The set of slots of a frame is denoted by $T = \{1, 2, \ldots, |T|\}$. The k-th slot of a link e is referred to as t_e^k. A slot is an atomic reservable bandwidth unit. We say link e_1 *conflicts with* e_2, or $e_1 \nparallel e_2$, if they share a common sender or receiver. Notice that, by this definition, if $e_1 = e_2$, then $e_1 \nparallel e_2$. Slot $t_{e_1}^{k_1}$ conflicts with $t_{e_2}^{k_2}$, if $k_1 = k_2$ and $e_1 \nparallel e_2$. Each individual slot can be in one of the following three states: *allocated*, *occupied*, or *free*. An allocated slot is one that is reserved to carry a unit of flow traffic. An occupied slot is one that an allocated slot conflicts with, hence cannot be used for data transmission. A free slot is one that is neither allocated nor occupied. The available bandwidth of a link is the set of free slots on that link. A slot is *consumed* if it is either allocated or occupied.

The state S of a network G is a triple (A, O, F) where the function $A : E \to T$ specifies the set of allocated slots on each link. The sets of occupied and free slots are specified by the functions O and F respectively. A state S is *conflict free* if no two allocated slots conflict with each other. The functions A, O, and F are obviously

related. Given G and $\nparallel$, O and F can be derived from A. Therefore, only A is the essential component of a conflict free state S. The total number of free slots,

$$\sum_{e \in E} |F(e)|,$$

characterizes the available network bandwidth that can be allocated to support users' requests.

The goal of our work is to design algorithms to process users' *flow request*. Let $s \xrightarrow{b} d$ denote a flow request between source s and sink d asking for b free slots. An $s \xrightarrow{1} d$ request is called a *unit request*.

To answer an end-to-end flow request, we restrict ourselves to solutions based on a single path. In particular, an $s \xrightarrow{b} d$ *flow arrangement* A_p is specified by an $s \rightarrow d$ path p, and b time slots to be allocated on each link e of p. The set of the b slots to be allocated on the link $e \in p$ is denoted by $A_p(e)$. We usually speak of a flow arrangement with respect to a state of the network. A flow arrangement A_p is *feasible* with regard to state $S = (A, O, F)$ if $A_p(e) \subseteq F(e)$ for every link e of p. That is, the time slots to be allocated by A_p are all free slots. Notice that, if S is conflict free and A_p is feasible, then the new network state $S' = (A', O', F')$ after the allocation of A_p is also conflict free. Here, A' is defined by $A'(e) = A(e) \cup A_p(e)$ for every $e \in p$, and $A'(e) = A(e)$ for every $e \notin p$. Recall that, O' and F' can be easily derived from A'. Unless explicitly declared, only feasible arrangements are considered in our discussion.

Given a state S, the *consumption set* $C(S, t_e^k)$ of a free slot t_e^k is defined to be $\{t_{e'}^k | t_{e'}^k \in F(e'), e \nparallel e'\}$, that is, the set of free slots that t_e^k conflicts with. The consumption set of a flow arrangement A_p, denoted by $C(S, A_p)$ is defined as $\bigcup_{e \in p} C(S, A_p(e))$, where $C(S, A_p(e)) = \bigcup_{t_e^k \in A_p(e)} C(S, t_e^k)$ is the set of free slots to be consumed by the allocation of $A_p(e)$. Note that, $t_e^k \in C(S, t_e^k)$ and $A_p(e) \subseteq C(S, A_p(e))$ by the above

definition. When S is understood, $C(S, A_p(e))$ and $C(S, A_p)$ will be abbreviated as $C(A_p(e))$ and $C(A_p)$. With the above definitions and notations, we formulate the end-to-end on-demand bandwidth allocation problem as follows. Given a network G, its state S and a flow request $s \xrightarrow{b} d$, find a feasible flow arrangement A_p if there is any, such that $|C(S, A_p)|$, the number of slots to be consumed by A_p, is minimized.

The above problem formulation essentially defines an optimal flow arrangement as the one that consumes the minimum amount of available bandwidth. We remark that, there are many other ways to define the optimality of a flow arrangement. For example, the optimal flow arrangement can be defined as the one such that after the allocation of the arrangement, the residual network has the maximum average available slots per link. Further discussions on other notions of optimal flow arrangements and the corresponding flow arrangement algorithms fall out of the scope of our study, and is left as one of the future work.

2. Complexity analysis

Before presenting our algorithm for the end-to-end bandwidth allocation problem defined in the previous section, we first study its complexity. The main result of this section is the following theorem.

Theorem 8. *The end-to-end bandwidth allocation problem with the single transceiver schedulability constraint is NP-hard.*

Proof. We consider the decision version of the problem, which asks for the existence of a flow arrangement that consumes less than a specific number of slots. The idea of this proof is to show that 3SAT is reducible to (the decision version of) the bandwidth allocation problem.

The 3SAT problem asks for the satisfiability of a given Boolean formula f in

the CNF form, that is, f is given as the conjunction of a number of clauses: $f = f_1 \wedge f_2 \wedge \ldots \wedge f_i \wedge \ldots$, where each clause f_i is the disjunction of up to three literals (Boolean variables or their negations). We use $|f|$ to denote the number of clauses in the formula f and $|f_i|$ to denote the number of literals in the clause f_i. We will use the following formula

$$f = f_1 \wedge f_2 \wedge f_3 = (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)$$

as an illustrating example throughout this proof.

For a given Boolean formula f, we can construct a corresponding directed graph G_f in the following two steps.

Step 1. For each clause f_i, we create a diamond shaped subgraph G_{f_i} with $|f_i|$ parallel paths corresponding to the $|f_i|$ literals in f_i. After that, all the $|f|$ constructed diamond subgraphs are connected one by one serially as shown in Fig 13, where the leftmost and rightmost nodes are named s and d respectively. All the links created in step 1 are called forward links, directed from s to d. The graph node corresponding to a particular literal $l \in f$ is denoted as v_l.

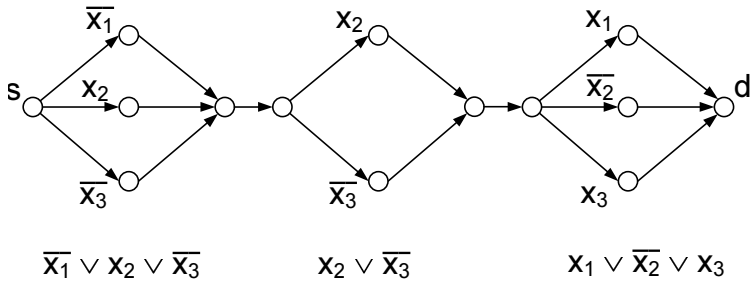


Fig. 13. Hardness: Step 1. The construction of a network with a sequence of diamond shaped subgraphs with only forwarding links.

Step 2. We add additional links to the graph constructed in step 1 as follows.

$\forall 1 \leq i < j \leq |f|$, and $\forall l_i \in f_i$ and $\forall l_j \in f_j$, we consider two cases:

Case 1: $l_i \not\equiv \overline{l_j}$. That is, l_i and l_j correspond to different Boolean variables, or they correspond to the same Boolean variable and both are in its original form or negation form. In this case, we add a directed link from node v_{l_j} to node v_{l_i}. In our example, if l_i is $\overline{x_1}$ in f_1 and l_j is x_2 in f_2, then the added link is the top link in Fig 14. If l_i is $\overline{x_3}$ in f_1 and l_j is $\overline{x_3}$ in f_2, then the added link is the bottom one. Links created in this case are called backward links.

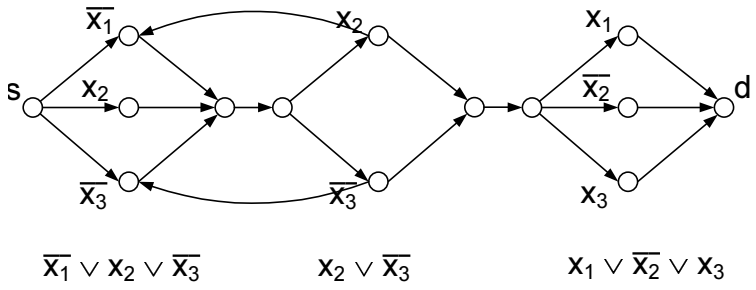


Fig. 14. Hardness: Step 2.1. The construction of backward links.

Case 2: $l_i \equiv \overline{l_j}$. That is, l_i and l_j correspond to the same Boolean variable, and they are the negation of each other. In this case, we add two extra nodes and links to the graph as follows: first, we add an extra node and a directed link from the extra node to node v_{l_i}; then, we add another extra node and a directed link to it from node v_{l_j}. In our example, suppose l_i is $\overline{x_3}$ in f_2 and l_j is x_3 in f_3, then we add the two nodes and two directed links as shown in Fig 15. Links created in this case are called dangling links.

After the two steps, we get a directed graph G_f as shown in Fig 16. Consider the network corresponding to G_f. Suppose each time frame contains only one slot, all slots are free, and an $s \xrightarrow{1} d$ request is to be established. Since $|T| = 1$, an $s \xrightarrow{1} d$ flow

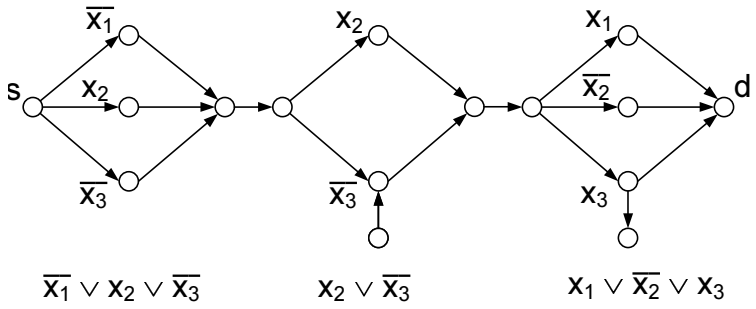


Fig. 15. Hardness: Step 2.1. The construction of dangling links.

arrangement A_p degenerates to path p. In other words, establishing a unit flow only involves routing in this setting. Any $s \xrightarrow{1} d$ path makes $3|f| - 1$ links be allocated. It also makes $\sum_{i=1}^{|f|} 2(|f_i| - 1)$ forward links created in step 1 be occupied, and Q number of backward and dangling links created in step 2 be occupied. To find an optimal path that consumes minimum number of links, we would like to minimize Q.

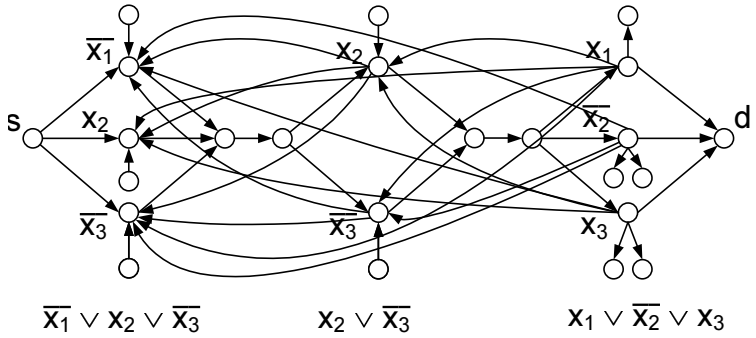


Fig. 16. The complete constructed network corresponding to Boolean formula $(\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)$.

Now we show a relationship between a solution to 3SAT and a solution to our bandwidth allocation problem. Specifically, we show Boolean formula f is satisfiable

if and only if there exists an $s \to d$ path p in G_f which consumes at most

$$(3|f|-1)+\sum_{i=1}^{|f|}2(|f_i|-1)+\sum_{i=1}^{|f|-1}\sum_{j=i+1}^{|f|}|f_j|+\sum_{i=2}^{|f|}\sum_{j=1}^{i-1}|f_j|-\binom{|f|}{2}$$

links, that is, if and only if for path p,

$$Q \le \sum_{i=1}^{|f|-1}\sum_{j=i+1}^{|f|}|f_j|+\sum_{i=2}^{|f|}\sum_{j=1}^{i-1}|f_j|-\binom{|f|}{2}. \qquad (*)$$

If 3SAT has a satisfying solution, then we can assign values to the Boolean variables such that for every $1 \le i \le |f|$, there is a literal $l_i \in f_i$ such that the assignment makes $l_i = true$. In G_f, consider the $s \to d$ path p that travels through v_{l_i}, $1 \le i \le |f|$. For v_{l_i}, attached to it there are $\sum_{j=i+1}^{|f|}|f_j|$ inbound links(either backward links or dangling links) and $\sum_{j=1}^{i-1}|f_j|$ outbound links created in Step 2. Therefore, the number of backward and dangling links to be consumed by path p is

$$\sum_{i=1}^{|f|-1}\sum_{j=i+1}^{|f|}|f_j|+\sum_{i=2}^{|f|}\sum_{j=1}^{i-1}|f_j|.$$

Notice that, some of the links are counted twice in the above enumeration. In fact, for any $1 \le i < j \le |f|$, by the construction of path p, we have $l_i \not\equiv \overline{l_j}$, and there is a link from v_{l_j} to node v_{l_i}. This link is considered as both an inbound link of v_{l_i} and an outbound link of v_{l_j}. Since there are $|f|$ clauses, there are in total $\binom{|f|}{2}$ such doubly counted links, and therefore, the actual number of backward and dangling links to be consumed by path p is

$$Q = \sum_{i=1}^{|f|-1}\sum_{j=i+1}^{|f|}|f_j|+\sum_{i=2}^{|f|}\sum_{j=1}^{i-1}|f_j|-\binom{|f|}{2},$$

i.e., (*) holds.

Now consider the other direction. Assume there is an optimal path p such that for this path, (*) holds. Suppose for $i = 1, 2, \cdots, |f|$, path p travels through v_{l_i},

where $l_i \in f_i$. By essentially the same reason as above, we can see that there cannot exist two nodes v_{l_i} and v_{l_j} with $1 \leq i < j \leq |f|$ on p such that $l_i \equiv \overline{l_j}$, otherwise, (*) cannot be satisfied. So, for any Boolean variable x, if it appears in one of the literals $l_1, l_2, \cdots, l_{|f|}$, it either appears only in the form of x or only in the form of $\overline{x}$. We assign values to every Boolean variable x in f as follows: among the literals $l_1, l_2, \cdots, l_{|f|}$, if x appears in the form of x, then we let $x = true$; if x appears in the form of $\overline{x}$, then we let $x = false$; otherwise, we let x take any Boolean value. Such an assignment must be a satisfying solution to the 3SAT problem.

In summary, 3SAT can be reduced to a special case of our end-to-end on-demand bandwidth allocation problem with single transceiver constraint in two steps. Apparently the reduction takes only polynomial time, that is to say, the bandwidth allocation problem is NP-hard. □

3. Minimum consumption routing and scheduling

In this section, we present the design of our end-to-end on-demand bandwidth allocation algorithm MCRS, and analyze its performance.

An illustrative example

Before getting into the detailed discussion of MCRS, we first use a toy network in Figure 17 to illustrate the basic idea of its routing sub-algorithm. To simplify the discussion, we assume that each frame has only one slot, so that establishing a unit flow connection only involves making routing decisions. That is, a flow arrangement A_p is uniquely specified by the path p.

Consider the network in Figure 17. Initially, all links are available. Suppose a unit flow between s and d is to be established. Two obvious routing choices are: *scd* and *suvd*, of which *scd* is the one with minimum number of hops. If *scd* is chosen,

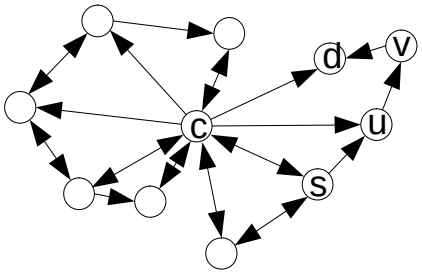
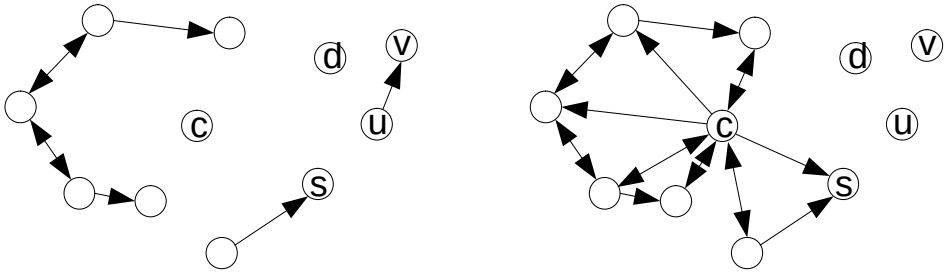


Fig. 17. A simple network illustrating the intuition of the MCSR algorithm.

then the link *sc* and *cd* will be allocated, and all the links in $E_{out}(s)$, $E_{in}(c)$, $E_{out}(c)$ and $E_{in}(d)$ are to be occupied. The residual network after the allocation of *sc* and *cd* is shown in Figure 18(a), where both *sc* and *cd*, as well as the links conflicting with them are left out. Similarly, the residual network after the allocation of *suvd* is shown in Figure 18(b). Clearly, network (a) is "less connected" than (b), and hence has a poorer chance to support future requests.



(a) path *scd* is chosen (b) path *suvd* is chosen

Fig. 18. Residual networks after the allocation of the paths *scd* and *suvd*.

This simple example shows, unlike in wired networks, choosing the path with minimum hops, *scd* in this case, does not necessarily lead to minimum bandwidth consumption in wireless network. It seems that a better strategy would be to avoid

choosing nodes with high degree, such as the node c in Figure 17. It is worth to point out that, the degree of a node is a dynamic quantity, which keeps changing as flow requests are processed. For example, the degrees of c in Figure 17 and Figure 18 are different. That is, the degrees of c before and after the flow request is processed are different. The same observation can also be made from the perspective of links: choosing a link $e = uv$ would result in $|E_{out}(u) \cup E_{in}(v)|$ links to be consumed, thus one would like to avoid choosing links with high $|E_{out}(u) \cup E_{in}(v)|$ value in order to reduce bandwidth consumption.

The above observation is based on the assumption that each frame has only one slot. Algorithm MCRS in the next subsection further develops this idea for the general case where a frame has more than one allocatable slots.

Minimum consumption routing and scheduling

Algorithm MCRS relies on two concepts, the b-th bottom set and b-th consumption level of a link e. The b-th *bottom set* of a link e, denoted by $B_b(S, e)$, is the set of b free slots of e with minimum consumption sets if $|F(e)| \geq b$, or Φ if $|F(e)| < b$.

The *consumption level* of a free slot t_e^k, denoted by $c(S, t_e^k)$, is the size of its consumption set $|C(S, t_e^k)|$. The b-th consumption level of link e, denoted by $c_b(S, e)$, is defined as $|C(S, B_b(e))|$ if $|F(e)| \geq b$, or ∞ if $|F(e)| < b$. In other words, the b-th consumption level of a link $e = uv$ is the minimum number of slots that will be consumed to support a $u \xrightarrow{b} v$ flow.

When S is understood, $B_b(S, e)$, $c(S, t_e^k)$, and $c_b(S, e)$ will be abbreviated as $B_b(e)$, $c(t_e^k)$, and $c_b(e)$. With these definitions, we present our routing and scheduling algorithm, MCRS, as shown in Figure 19.

Basically, MCRS uses the b-th consumption level for routing (step 1 and 2) and b-th bottom set for scheduling (step 4 and 5). This method of routing and scheduling

Input: network graph $G = (V, E)$, state $S = (A, O, F)$, flow request $s \xrightarrow{b} d$

Output: a feasible $s \xrightarrow{b} d$ flow arrangement A_p if there is one

1. For each $e \in E$, compute the b-th consumption level $c_b(e)$
2. Use $c_b(e)$ as the cost of $e \in E$ to find a shortest $s \rightarrow d$ path p by Bellman-Ford's algorithm
3. If p does not exist or the cost of p is ∞, then return "failed", else
4. For each link $e \in p$, compute the b-th bottom set $B_b(e)$
5. Let $A_p(e) = B_b(e)$, and return A_p

Fig. 19. MCRS, the Minimum Consumption Routing and Scheduling algorithm.

will hereafter be called MCR (minimum consumption routing) and MCS (minimum consumption scheduling) respectively. If the algorithm does not return "failed" in step 3, it returns a flow arrangement A_p. The returned arrangement A_p is obviously feasible since $B_b(e) \subseteq F(e), \forall e \in p$. Accordingly, the new network state after the allocation of A_p is conflict free.

The running time of algorithm MCRS is dominated by step 1, 2 and 4. Step 1 computes the b-th consumption level for each link $e \in E$, which can be done in time $O(b|E||T|)$ in the following two substeps. In the first substep, we compute the consumption level of every free slot as follows. For a free slot t_e^k, its consumption level $c(t_e^k) = c_{in}(t_e^k) + c_{out}(t_e^k) + 1$. Here, $c_{in}(t_e^k)$ and $c_{out}(t_e^k)$ are the number of free slots $t_{e'}^k$, $e \neq e'$, such that e and e' share the same receiver and sender respectively. The number "1" corresponds to slot t_e^k itself. Notice that, slots t_e^k and $t_{e'}^k$ have the same c_{in} value if e and e' share the same receiver. Based on this fact, the c_{in} value for all

the free slots can be computed in time $O(|E||T|)$. Similarly, the c_{out} value and hence the consumption level for all the free slots can also be computed in time $O(|E||T|)$. Then in the second substep, we spend another $O(b|T|)$ time for each link $e \in E$, that is, $O(b|E||T|)$ for all the links, to find the b slots with minimum consumption levels for each link and compute the b-th consumption level. Therefore, step 1 takes time $O(b|E||T|)$ in total. Step 2 runs Bellman-Ford's shortest path algorithm which takes time $O(|E||V|)$. Step 4 computes the minimum consumption schedule for every link of the path found by step 2. By an analysis similar to that of step 1, step 4 takes time $O(b|E||T|)$. In summary, Algorithm MCRS takes time $O((|V| + b|T|) \cdot |E|)$.

The NP-hardness of the bandwidth allocation problem with single transceiver constraint shows that an optimal arrangement cannot be found by an polynomial time algorithm, assuming P is not equal to NP. Nevertheless, algorithm MCRS provides a solution whose cost is guaranteed to be no more than twice the cost of an optimal solution.

Theorem 9. *MCRS is a 2-approximation algorithm.*

Proof. Let A_q be any $s \xrightarrow{b} d$ flow arrangement where q is an arbitrary $s \rightarrow d$ path. We first show,

$$|C(A_q)| \leq \sum_{e \in q} |C(A_q(e))| \leq 2|C(A_q)| \tag{1}$$

Before showing (1) is true, we need to introduce the concepts of "singly consumed" and "doubly consumed" slots. We say a free slot is to be singly consumed by A_q if it belongs to the consumption set of exactly one free slot to be allocated by A_q. A free slot is to be doubly consumed if it belongs to the consumption sets of exactly two free slots to be allocated. Since any link e has at most two ends on path q, slot t_e^k is either to be singly or doubly consumed by A_q. For example, Fig 20 shows a fraction of a network, where each frame has three slots, and each currently consumed slot is

marked with a "x". Suppose a $u \xrightarrow{2} v$ flow is to be established, which allocates t^2_{uw}, t^3_{uw}, t^1_{wv}, and t^3_{wv}. Then, t^3_{uv} is to be doubly consumed by this arrangement because $t^3_{uv} \in C(t^3_{uw})$ and $t^3_{uv} \in C(t^3_{wv})$, and t^2_{uw} and t^2_{ux} are to be singly consumed because they are only in the consumption set of t^2_{uw}.

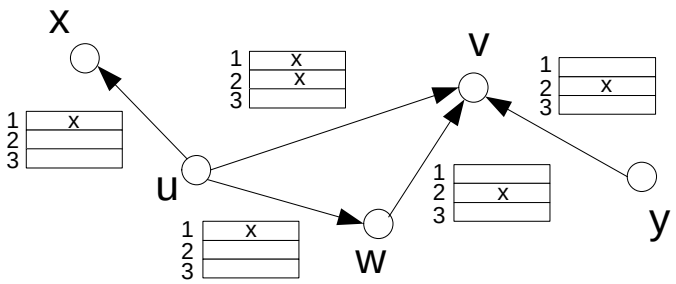


Fig. 20. Doubly and singly consumed slot.

For an arrangement A_q, let S_1 and S_2 respectively denote the set of slots to be singly and doubly consumed. Then $|C(A_q)| = |S_1| + |S_2|$ and $\sum_{e \in q} |C(A_q(e))| = |S_1| + 2|S_2|$, and (1) follows.

Now, let A_p be the arrangement outputted by algorithm MCRS, and A_q be an optimal flow arrangement. By (1),

$$|C(A_p)| \leq \sum_{e \in p} |C(A_p(e))|. \tag{2}$$

By algorithm MCRS and the definition of $c_b(e)$, $A_p(e) = B_b(e)$ and $c_b(e) = |C(B_b(e))|$ for every $e \in p$, so,

$$\sum_{e \in p} |C(A_p(e))| = \sum_{e \in p} |C(B_b(e))| = \sum_{e \in p} c_b(e). \tag{3}$$

A_p is the shortest path using $c_b(\cdot)$ as the cost function, thus,

$$\sum_{e \in p} c_b(e) \leq \sum_{e \in q} c_b(e). \tag{4}$$

By the definition of $c_b(\cdot)$, $\forall e \in q$, $c_b(e) \leq |C(A_q(e))|$, so,

$$\sum_{e \in q} c_b(e) \leq \sum_{e \in q} |C(A_q(e))|. \tag{5}$$

By (2), (3), (4), (5), and (1), we have

$$|C(A_p)| \leq 2|C(A_q)|. \tag{6}$$

That is, the amount of bandwidth to be consumed by A_p is no more than twice the amount of bandwidth to be consumed by the optimal flow arrangement, which concludes our proof that MCRS is a 2-approximation algorithm. □

Theorem 9 analyzes the theoretical bound on the performance of MCRS. In practice, the performance of MCRS may be significantly better. In particular, MCRS consumes twice the amount of bandwidth the optimal flow arrangement would consume if and only if inequality (2), (4), (5), and (1) all hold with equality. A thorough study of the likelihood that equality holds for (2), (4), (5), and (1), falls out of the scope of our discussion, and we take it as future research.

Stateless and semi-stateless variations

As one of the two components of MCRS, MCS is amenable to efficient distributed implementation. The main task of MCS is to compute $B_b(e)$ for every link $e = uv$ of the path p chosen by MCR. To do this, we can simply let u collect the state of links in $E_{out}(u) \cup E_{in}(v)$, and compute $B_b(e)$. In this way, the computation of MCS can be performed concurrently using only local information.

MCR, based on Bellman-Ford's algorithm, on the other hand, is relatively harder to implement in an efficient way. A common method of implementing Bellman-Ford's algorithm is to precompute and cache routing decisions in a table. This method may not be suitable for MCR because it requires up-to-date network state information, which may change frequently as flow requests arrive and leave.

To address the above issue, we propose the following stateless version of MCR, called MCR$^-$. Specifically, in MCR$^-$ we use $|E_{out}(u) \cup E_{in}(v)|$ for every $e = uv \in E$ instead of $c_b(e)$ as the cost function for routing. This idea of MCR$^-$ has actually already been illustrated in the example at the beginning of this section. The intuition is essentially the same as that of MCR, that is, to avoid choosing "highly connected" nodes as much as possible. Note that, MCR$^-$ only requires the network topology as input, that is, it needs not to be aware of the current network state. Therefore, routing decisions can be precomputed efficiently by the distributed Bellman-Ford's algorithm.

Routing can also be done in a semi-stateless manner where the cost function for routing is defined as follows: if $|F(e)| \geq b$, then the cost of $e = uv$ is $|E_{out}(u) \cup E_{in}(v)|$, otherwise, the cost is ∞. This semi-stateless MCR needs $|F(e)|$ as input, $\forall e \in E$, that is, it does need to be aware of the information of available bandwidth for each network link. With this information, semi-stateless MCR can avoid choosing routes with insufficient available bandwidth, and make more accurate decisions than completely stateless MCR$^-$. Here, $|F(e)|$ is only a single number, which is significantly simpler than $S(e) = (A(e), O(e), F(e))$, what MCR needs to know. As a result, semi-stateless MCR is expected to be more efficient than MCR in terms of computation and communication overhead.

In summary, stateless and semi-stateless routing algorithms trade bandwidth efficiency for computation and communication efficiency. Stateless and semi-stateless

algorithms do not provide any upper bound on the amount of bandwidth to be consumed. The actual performance of stateless algorithm will be experimentally studied in Section 4.

4. Simulation evaluation

To evaluate the performance of our scheme, we perform a series of simulations on a experiment network with 200 nodes randomly distributed in a 500×500 rectangular plane area. For each node, we assign a random (Gaussian) transmission range with mean and variance set to 100 and 50. A node can communicate uni-directionally with any node within its transmission range. The number of slots per frame is set to be 50. Note, although this simulation setting implies a disk graph based network model, MCRS is designed for general graph models. We take this simulation design mainly because it is a widely adopted set up, and we use it as a benchmark to test performance.

To the our best knowledge, no other non-trivial routing and scheduling algorithm has been proposed in the literature. Thus, in our experiments, we fix the scheduling algorithm to be MCS, and compare the performance of MCR and MHR, and their stateless versions, MCR$^-$ and MHR$^-$. Here, MHR is acronym for minimum hop routing, a widely used routing algorithm for networks without schedulability constraint. Note, when we speak of the performance of one of the four routing algorithms, we are really talking about the performance of its combination with MCS. For MHR, we assign a cost of 1 (one hop) to link e, $\forall e \in E$, if $|F(e)| \leq b$, or ∞ if $|F(e)| > b$. For MHR$^-$, we simply let the cost of each link to be 1 regardless of the current network state. For convenience, MCR and MCR$^-$ will be collectively called min-consumption algorithms, and MHR and MHR$^-$ min-hop algorithms.

Static experiment

In the first experiment, we load the network with static requests. Once a static flow request is accepted and established, it stays in the network indefinitely, and the allocated bandwidth is not recycled. We load the network with 5000 such unit requests one by one, each of which is assigned a random source and sink. We observe the bandwidth and delay performance, as well as load balancing performance for each algorithm.

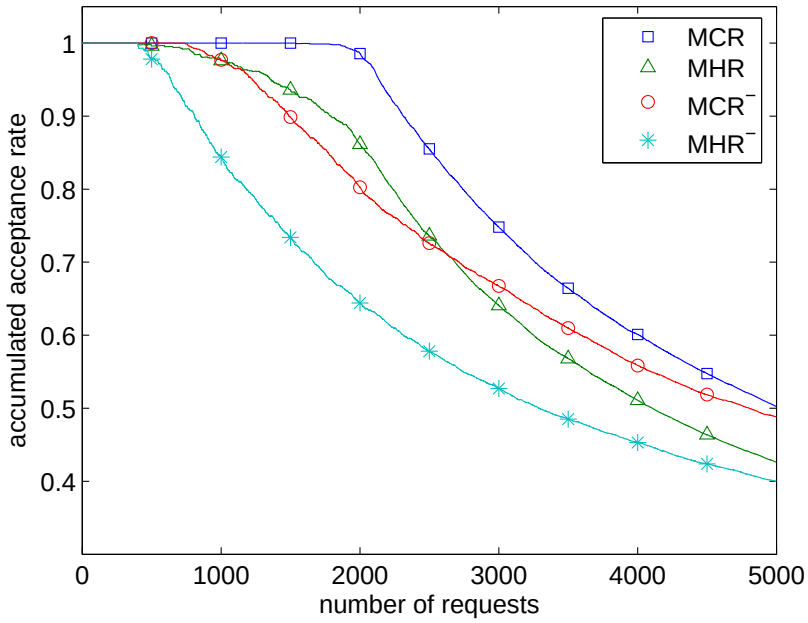


Fig. 21. Accumulated acceptance rate. The x axis represents the number of loaded requests, and the y axis represents the number of requests accepted by each algorithm.

Figure 21 plots the accumulated acceptance rate, y, after x requests are loaded, $0 < x \leq 5000$, that is, $y = a(x)/x$, where $a(x)$ is the number of accepted requests out

of the x loaded ones. As shown in this figure, MCR maintains a 100% acceptance rate up to 1537 requests, while this number for MHR, MCR$^-$, and MHR$^-$ is 406, 742, and 401 respectively. We take the point x up to which an algorithm maintains a 100% acceptance rate as a simple measurement of that algorithm's bandwidth performance. By this measurement, the bandwidth performance of MCR is more than 4 times of that of MHR in this experiment (in all our other experiments, the bandwidth performance of MCR is consistently at least 3 times that of MHR). It is interesting to notice that the stateless algorithm MCR$^-$ exhibits better performance than the stateful algorithm MHR. In Figure 21, both y_{MCR} and y_{MHR} keep dropping monotonously when more and more requests are loaded, but y_{MCR} is consistently higher than y_{MHR}. In the end, MCR accepted 2511 out of 5000 requests, which is 17.9% more than MHR. y_{MCR^-} and y_{MHR^-} exhibit the same pattern. Interestingly, y_{MCR^-} falls below y_{MHR} at $x \approx 1000$, but it ends up to be higher than y_{MHR} when x approaches 5000.

Figure 22 shows the average number of free slots per link, y, after x requests are loaded, $0 < x \leq 5000$. As expected, $y_{\text{MCR}} > y_{\text{MCR}^-} > y_{\text{MHR}} = y_{\text{MHR}^-}$ when x is small, which shows min-consumption algorithms consumes less bandwidth than min-hop algorithms when the same amount of requests are accepted. Here, $y_{\text{MHR}} = y_{\text{MHR}^-}$ because MHR and MHR$^-$ choose different paths only after certain shortest paths are saturated. Notice that, $y_{\text{MCR}} < y_{\text{MHR}} < y_{\text{MCR}^-} < y_{\text{MHR}^-}$ when x approaches 5000. This is not surprising because as shown in Figure 21, min-consumption algorithms have accepted more requests than min-hop algorithms at this moment. y_{MCR} has an abrupt turn at $x \approx 2000$. Before this point, y_{MCR} decreases quickly and almost linearly as requests are loaded one by one, and after this point, y_{MCR} begins to decrease at a very low speed. Similar point can be observed for y_{MHR} at $x \approx 1900$, although the turn is less abrupt than that of y_{MCR}. We call this point the saturation point

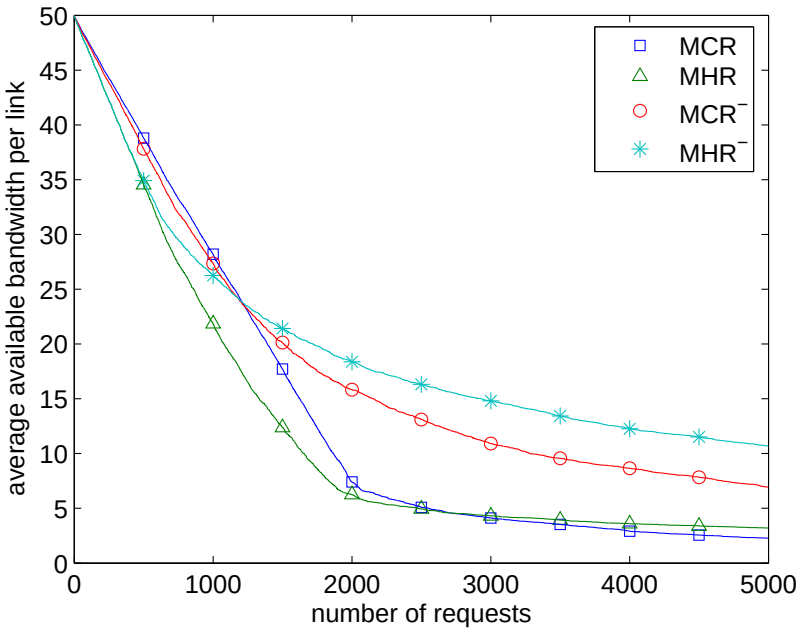


Fig. 22. Available bandwidth. The x axis represents the number of loaded requests, and the y axis represents the average number of free slots per link.

in that it roughly indicates the moment when the network bandwidth is exhausted. No obvious saturation points can be observed for y_{MCR^-} and y_{MHR^-}. The existence of saturation point for stateful algorithms shows that they are more aggressive at bandwidth allocation than stateless algorithms.

Next, we evaluate the delay performance of each algorithm, where delay is measured by the number of hops of the path computed for each accepted request. In Figure 23, We show the accumulated average hops, y, after x requests are loaded, where $y = \sum_{i=1}^{x} h(i)/a(x)$ with $h(i)$ and $a(x)$ defined as follows. If request i is accepted, then $h(i)$ is the number of hops of the path found by each routing algorithm; if request i is rejected, then $h(i) = 0$. $a(x)$ again is the number of accepted requests out of x loaded requests. At first when x is small, $y_{\text{MCR}} > y_{\text{MCR}^-} > y_{\text{MHR}} = y_{\text{MHR}^-}$. This indicates that min-hop based algorithms have better delay performance than

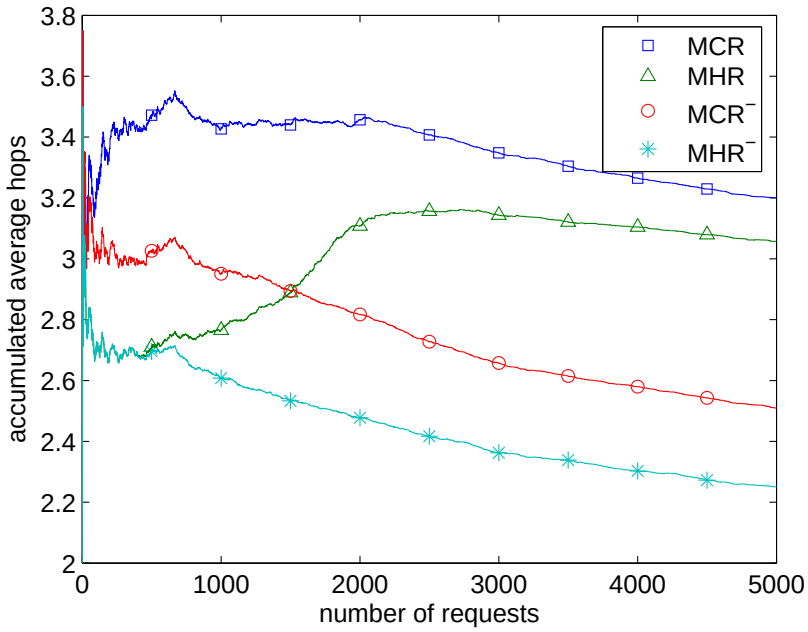
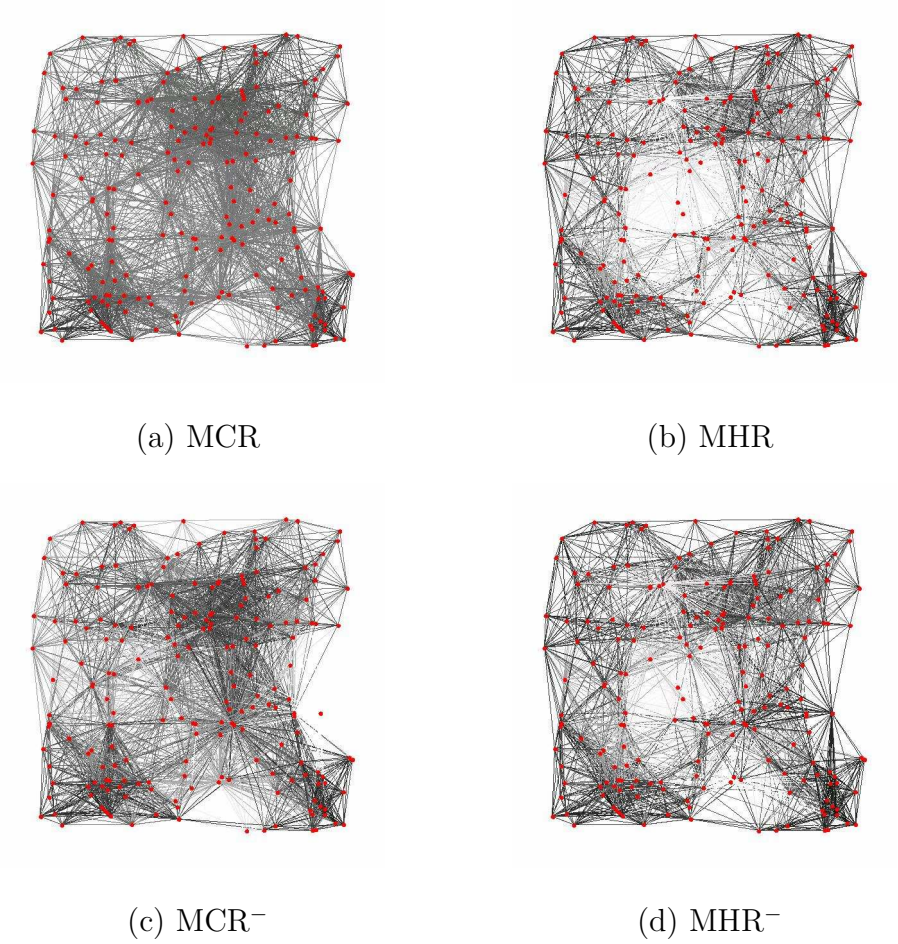


Fig. 23. Accumulated average hops. The x axis represents the number of loaded requests, and the y axis represents the average path length for each accepted request.

min-consumption based algorithms, which conforms to their design objective to reduce delay. Here, $y_{\text{MHR}} = y_{\text{MHR}^-}$ is again because MHR and MHR$^-$ make the same routing decision when no path is saturated. When more requests are loaded, more and more min-hop paths are saturated, and MHR is forced to pick sub-shortest paths. This is why y_{MHR} keeps increasing until it reaches a peak at the saturation point of $x \approx 1900$. After that, y_{MHR} begins to keep dropping, because when the network becomes more and more saturated, it is only capable of supporting shorter and shorter requests. In comparison, y_{MCR} maintains a value of around 3.4 up to the saturation point of $x \approx 2000$. After that, y_{MCR} starts to keep dropping (although still higher than y_{MHR}). The reason that y_{MCR} does not follow the climbing-descending pattern as that of y_{MHR} is because MCR is better at load balancing(to be further illustrated

shortly): when a short path is saturated, so are the longer ones. Finally, just as MHR versus MCR, the delay performance of MHR$^-$ is always better than that of MCR$^-$. No hill climbing is observed for both y_{MCR^-} and y_{MHR^-} because they are stateless: no longer paths are tried when shorter ones are saturated.



(a) MCR (b) MHR

(c) MCR$^-$ (d) MHR$^-$

Fig. 24. Residual networks after 1000 requests. The amount of available bandwidth of each link is indicated by the darkness of that link.

To study the load balancing abilities of min-consumption and min-hop algorithms, we examine the network for each algorithm after the first 1000 requests are

loaded, as shown in Figure 24. Here, 1000 is an arbitrarily chosen number. We performed the same studies for operation points other than 1000, and all the studies produced similar results. In Figure 24, we use darkness to indicate the amount of available bandwidth of each link. The darker a link is, the more bandwidth is available. From the figure we can see that the available bandwidth distributions of network (a) and (c) are more uniform than those of (b) and (d). This is because min-hop algorithms choose the shortest path for each request. Since the source and sink node of each request is random, the nodes sitting at the center of the network are more likely to be loaded first. This is not the case for min-consumption algorithms, however, because they do not particularly favor short paths.

Figure 25 plots the bandwidth distributions of the four networks in histograms. That is, this figure plots $y = d(x)$ for each algorithm, where $d(x)$ is the number of links with x available slots, $0 \leq x \leq 50$. In the four networks, the bandwidth spectrum of the MCR network is the narrowest. More accurately, the variances for MCR, MHR, MCR^-, and MHR^- are 4.16, 14.83, 9.80, and 14.15 respectively, which shows that the load balancing abilities of min-consumption algorithms are significantly better than min-hop algorithms.

Mixed experiment

In this experiment, we consider both static and dynamic requests. Dynamic flows have limited life-time. After they expire, the allocated bandwidth can be recycled to support other new flows. The arrival time of each dynamic request is modeled as a Poisson process, where the average inter-arrival time is set to 0.5 second. The life-time of a dynamic flow is modeled as a random variable with uniform distribution from 0 to 5 seconds. We first load the network with 1000 static unit requests, and then another 4000 dynamic unit requests. We perform 15 such experiments, each

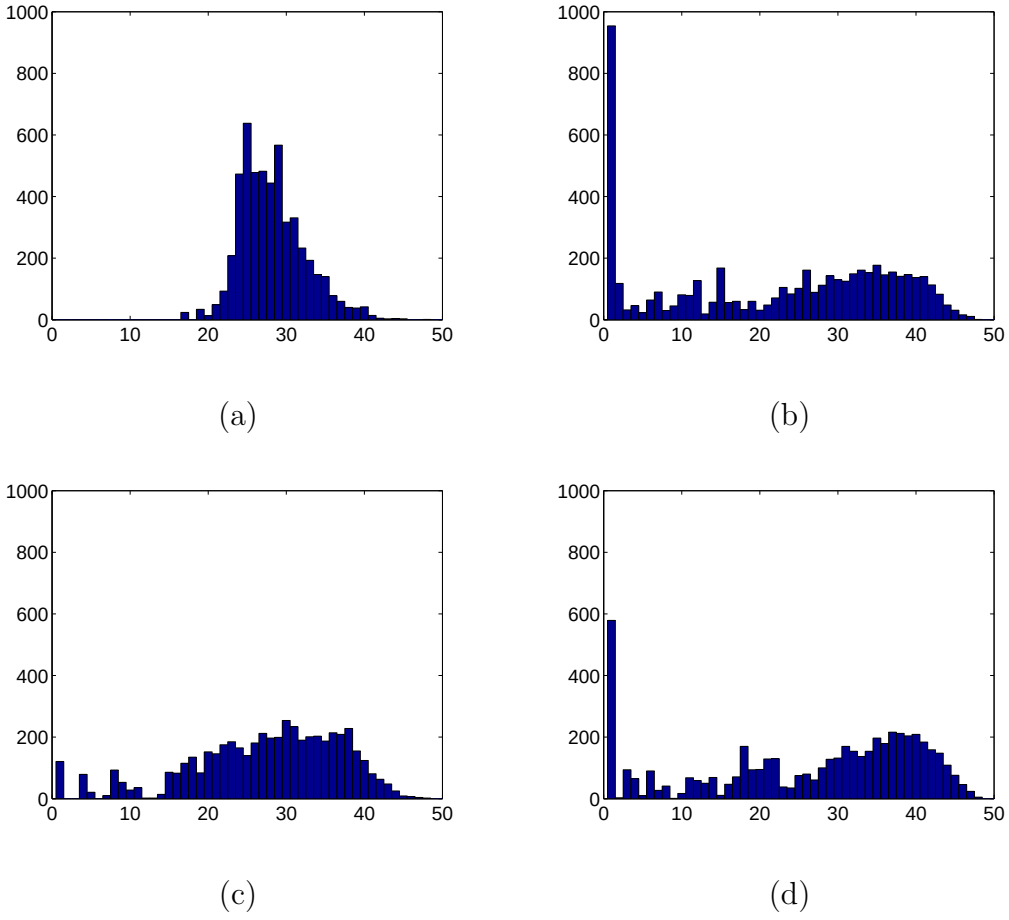


Fig. 25. Distribution of available link bandwidth after 1000 requests. y is the number of links with x available slots.

with a randomly generated network with the same parameters as those of the static experiment, and a set of mixed static and dynamic requests. Figure 26(a) shows the acceptance ratio of the 4000 dynamic requests for each of the 15 experiments. Figure 26(b) shows the average number of hops for the 4000 dynamic requests for each of the 15 experiments. Results confirm again that min-consumption algorithms have better bandwidth performance while min-hop algorithms have better delay performance.

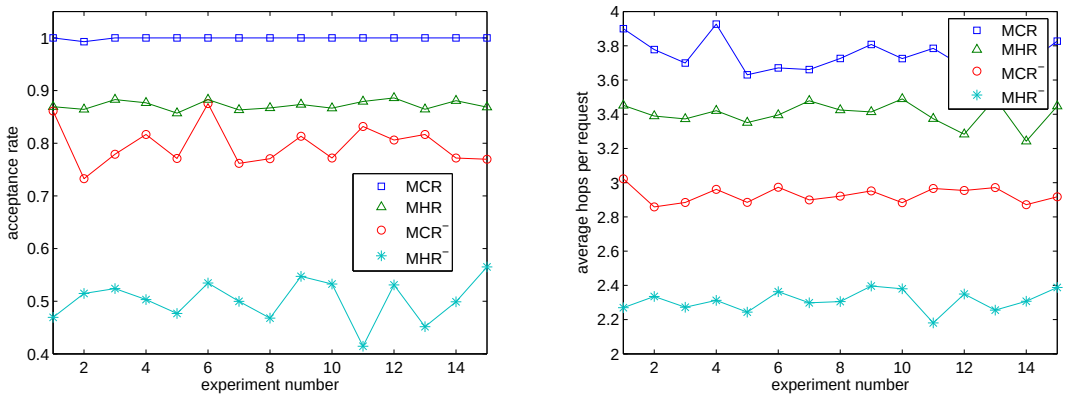


Fig. 26. Acceptance rate for the mixed experiments where the network is loaded with 1000 static and 4000 dynamic requests.

D. Throughput estimation

In this section, we study the problem of estimating the end-to-end throughput for wireless ad hoc networks with radio interference, where the term throughput refers to the maximum amount of data that can be successfully delivered across the network from a source node to a sink in a given period of time. End-to-end throughput is a fundamental network parameter that can be used to identify potential bottleneck, coordinate network traffic, plan and evaluate network design, etc. It is generally difficult to compute the exact end-to-end throughput under various schedulability constraints, and also, it is often satisfactory to have a reasonably small range within which the thoughput falls in. Because of this, in this section, we will only focus on efficient methods that give a reasonably good upperbound of the thoughput. In particular, we show that, under the interference model described in [1], a practically tight upperbound can be obtained by identifying a small set of carefully chosen cliques.

1. Problem statement

The schedulability constraints considered in this section models the effect of radio interference in wireless networks. We adopt the interference model in [1]. [1] assumes that each node has a radio transceiver with a certain transmission range, and a transmission from u to v is successful if all the other nodes whose transmission range that v is within are not transmitting. This interference model can also be stated in terms of links as follows. Let $e = uv$ and $e' = wh$ be two links; $e \nparallel e'$, that is, e conflicts with e' at node v if $wv \in E$, $h = v$, or $w = v$, see Figure 27(a), (b), and (c); transmission on e is successful if all the links that e conflicts with are not in transmission. In addition to the above three interference patterns, transmissions on link e and e' shown in Figure 27(d) are also considered to conflict with each other. This is the single transceiver constraint that we have considered before. Traditionally, the interference pattern in Figure 27(a) is called the *secondary interference* [46], while the patterns in Figure 27(b), (c) and (d) are called the *primary interference*. For convenience, the graph $(E, \nparallel)$ is called the conflict graph. Take Figure 28 as an example, $25 \nparallel 47$ at node 5, $41 \nparallel 46$, $14 \nparallel 42$ and $14 \nparallel 64$ at node 4.

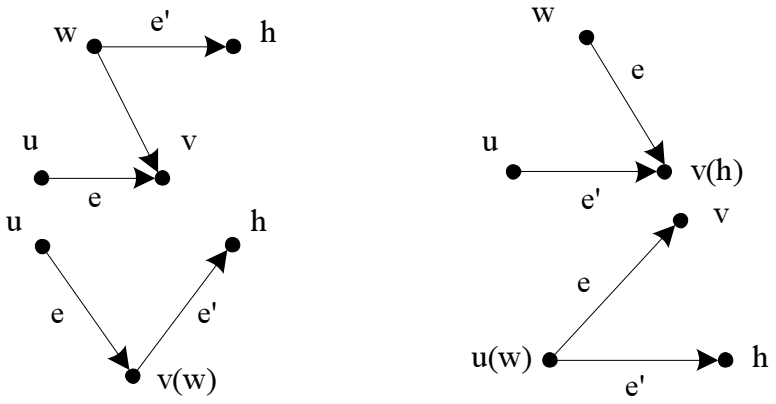


Fig. 27. The four patterns of the radio interference model from [1].

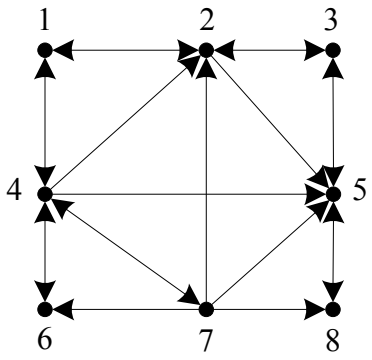


Fig. 28. A sample network. Each double arrowed line represents two one way links.

The notion of end-to-end throughput depends on the notion of schedule, which specifies the action of every link during one time unit. Specifically, a *schedule* is a 0-1 vector z indexed by $e \in E$ and $0 \leq t \leq 1$. $z_{e,t} = 1$ indicates that e is scheduled to be active at t, otherwise e is scheduled to be inactive. Schedule z is *feasible* if $\forall e, e' \in E$, $0 \leq t \leq 1$, $z_{e,t} = z_{e',t} = 1$ implies e does not conflict with e'. In other words, z is feasible if at any time $0 \leq t \leq 1$, the set of links scheduled to be active is conflict free.

Let c_e denote the capacity of link $e \in E$, that is, the maximum amount of data that can be transmitted by link e in unit time. Let s and d be a pair of source and sink nodes. An $s - d$ *flow* x is a vector indexed by $e \in E$ that satisfies the capacity constraints

$$0 \leq x_e \leq c_e, \forall e \in E,$$

and the flow conservation constraints

$$x(v) = 0, \forall v \in V - \{s, d\},$$

where $x(v) = \sum_{e \in E_{out}(v)} x_e - \sum_{e \in E_{in}(v)} x_e$. The capacity constraint simply says that the amount of data that link e carries cannot exceed its capacity. The flow conserva-

tion constraint says that the amount of data that flow into each node (except s and d) is equal to the amount of data that flow out of it. The *value* of an $s-d$ flow x is $x(s)$, the net amount of data that flows out of source s in unit time. Flow x is *schedulable* if there exists a feasible schedule z such that

$$x_e/c_e = \int_0^1 z_{e,t}dt, \forall e \in E.$$

If the above equation holds, then x is said to be realized by z. The term $\int_0^1 z_{e,t}dt$ here is the fraction of time that e is scheduled to be active.

With the above notions, we can now define the end-to-end throughput estimation problem as follows. Given a network and two nodes $s, d \in V$, the problem of end-to-end throughput is to find a schedulable $s-d$ flow x with maximum value. This value, denoted as θ_{sd}, is called the $s-d$ *throughput.*

The above formulation of the throughput problem with the schedulability constraint is a variant of the well known maxflow problem. The different between the two problems is, the throughput problem has an extra constraint that the flow must be schedulable (under the schedulability constraint) while the traditional maxflow problem does not have such restrictions. Due to this extra constraint, the feasible solution space of our throughput problem has a more complicated combinatorial structure. More specifically, it may not be possible to describe the feasible solution space by a small set of linear constraints. The key idea of the upperbounding approaches to be presented in the following discussion is to use a polynomial number of linear constraints to characterize a superset of the feasible solution space.

Note that the notion of a flow here is different from that in the last section. In particular, in this section, an end-to-end flow is delivered via multiple paths while a flow in the last section is restricted to a single path.

2. Upperbounding the end-to-end throughput

One approach to estimate the end-to-end throughput is based on the notion of *independent link set*, a set of mutually non-conflicting links. The *independence number* $\alpha(F)$ of a set of links $F \subseteq E$ is the size of the maximum independent set of F. Notice that if a flow x is schedulable, then it must satisfy the following *independence constraint*:

$$\sum_{e \in F} x_e/c_e \leq \alpha(F), \forall F \subseteq E.$$

To see that this necessary condition is true, let z be a feasible schedule that realizes x. By the definition of schedulability, $\forall e \in E$, $x_e/c_e = \int_0^1 z_{e,t} dt$. Thus,

$$\sum_{e \in F} x_e/c_e = \sum_{e \in F} \int_0^1 z_{e,t} dt = \int_0^1 \sum_{e \in F} z_{e,t} dt.$$

By the feasibility of z, the set of active links at time t is conflict free. In other words, the set of active links at time t forms an independent set. Since $\alpha(F)$ is the size of the maximum independent set of F, it must be true that $\sum_{e \in F} z_{e,t} \leq \alpha(F)$. And therefore, the independence constraint follows.

Since every schedulable flow satisfies the independence constraint, it together with the flow conservation constraint and the capacity constraint specifies a superset of the feasible solution space. It follows directly that, the optimal objective value $\bar{\theta}_{sd}(\mathcal{F})$ of the following linear program LP($\mathcal{F}$) is an upperbound of θ_{sd}, the end-to-end $s-d$ throughput, where $\mathcal{F} \subseteq 2^E$ is an arbitrary set of subsets of E:

$$
\begin{aligned}
\text{maximize} \quad & x(s) \\
\text{subject to} \quad & \sum_{e \in F} x_e / c_e \leq \alpha(F), && \forall F \in \mathcal{F} \\
& x(v) = 0, && \forall v \in V - \{s, d\} \\
& 0 \leq x_e \leq c_e, && \forall e \in E
\end{aligned}
$$

Although any $\mathcal{F} \subseteq 2^E$ corresponds to an upperbound $\bar{\theta}_{sd}(\mathcal{F})$ of θ_{sd}, the above linear program does not tell us which choice of $\mathcal{F}$ is good in the sense that the corresponding $\bar{\theta}_{sd}(\mathcal{F})$ is close to θ_{sd} and is easy to compute. In general, the more link sets $\mathcal{F}$ contains, the tighter $\bar{\theta}_{sd}(\mathcal{F})$ will be. The tightest $\bar{\theta}_{sd}(\mathcal{F})$ is achieved when $\mathcal{F} = 2^E$. $\bar{\theta}_{sd}(2^E)$, however, is obviously expensive to compute since there are exponential number of elements in 2^E. In the following, we propose several choices of good $\mathcal{F}$. In particular, we restrict our attention to cliques.

A *clique* is a set of mutually conflicting links. There are several advantages to deal with cliques instead of arbitrary link sets. First, given an arbitrary clique F, the term $\alpha(F)$ in LP($\mathcal{F}$) can be immediately substituted with 1 since the links in F mutually conflict with each other. This is in contrast to the fact that it is often expensive to compute $\alpha(F)$ for an arbitrary link set F. Second, given two class of cliques $\mathcal{F}_1$ and $\mathcal{F}_2$, we can often know which one of $\bar{\theta}_{sd}(\mathcal{F}_1)$ and $\bar{\theta}_{sd}(\mathcal{F}_2)$ is tighter without solving LP($\mathcal{F}_1$) and LP($\mathcal{F}_2$). Specifically, if for every clique $F_2 \in \mathcal{F}_2$, there is a clique $F_1 \in \mathcal{F}_1$ such that $F_2 \subseteq F_1$, then we say $\mathcal{F}_1$ is *stronger* than $\mathcal{F}_2$. If $\mathcal{F}_1$ is stronger than $\mathcal{F}_2$, then the independence constraints on $\mathcal{F}_2$ must be implied by the independence constraints on $\mathcal{F}_1$, and as a result $\bar{\theta}_{sd}(\mathcal{F}_1)$ must be closer to θ_{sd} than $\bar{\theta}_{sd}(\mathcal{F}_2)$. Therefore, in order to obtain a relatively tight upperbound $\bar{\theta}_{sd}(\mathcal{F})$ within a reasonable amount of time, one would want to pick a small set $\mathcal{F}$ of strong cliques .

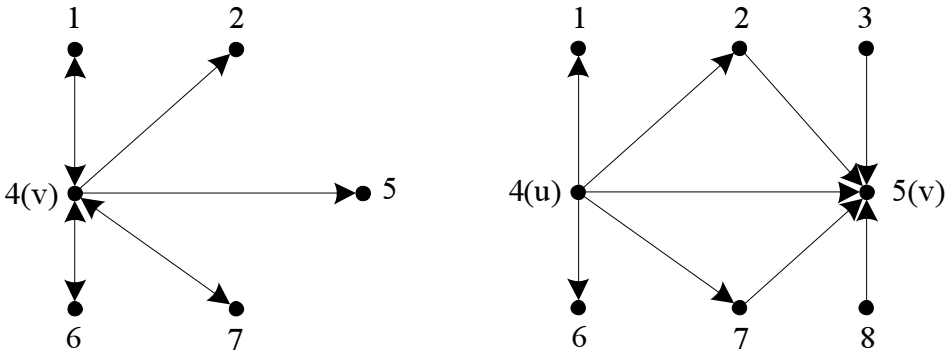


Fig. 29. Star and bell clique.

We start our discussion of choosing a good set of cliques by considering the following class of cliques call the star cliques:

$$\mathcal{Q}_s = \{E_{in}(v) \cup E_{out}(v), \forall v \in V\}. \tag{7}$$

By the primary interference, any two links in $E_{in}(v) \cup E_{out}(v)$ conflict with each other at the node v, thus each member of $\mathcal{Q}_s$ is indeed a clique. Figure 29(a) shows the star clique on node 4 of the network in Figure 28, which contains link 14, 41, 42, 45, 46, 64, 47, and 74. Another class of cliques, called the bell cliques, has the following form:

$$\mathcal{Q}_b = \{E_{out}(u) \cup E_{in}(v), \forall uv \in E\}. \tag{8}$$

We say $E_1 \subseteq E$ conflicts with $E_2 \subseteq E$, if $\forall e_1 \in E_1$ and $\forall e_2 \in E_2$, e_1 conflicts with e_2. Each member of $\mathcal{Q}_b$ is indeed a clique since $E_{out}(u)$ and $E_{in}(v)$ are both cliques by the primary interference and $E_{out}(u)$ conflicts with $E_{in}(v)$ at node v by the secondary interference. For example, the bell clique on link 45 of the network in Figure 28 is depicted in Figure 29(b).

By the definition of star and bell cliques, there are $|V|$ and $|E|$ number of star and bell cliques respectively for a given network, thus all of them can be enumerated

efficiently. Starting from star and bell cliques, we identify three classes of stronger cliques in the following discussion. To ease the exposition, we introduce the following additional notations. The set of predecessors and successors of a node v are denoted as $pred(v)$ and $succ(v)$. Define $pred[v] = pred(v) \cup \{v\}$ and $succ[v] = succ(v) \cup \{v\}$. Let U be a set of nodes, and v be a single node. The set of all links $e = uv$ such that $u \in U$ is denoted as $E(U \rightarrow v)$. Similarly, $E(v \rightarrow U)$ denotes the set of all links $e = vu$ such that $u \in U$.

First, consider the following class of cliques called class one cliques:

$$\mathcal{Q}_1 = \{E_{in}(u) \cup E_{out}(u) \cup E(pred(u) \rightarrow v), \forall uv \in E\}.$$

It is straightforward to verify that $E(pred(u) \rightarrow v)$ conflicts with $E_{in}(u)$ either at u or v, and $E(pred(u) \rightarrow v)$ conflicts with $E_{out}(u)$ at v. We have already known that $E_{in}(u) \cup E_{out}(u)$ forms a star clique, therefore, $E_{in}(u) \cup E_{out}(u) \cup E(pred(u) \rightarrow v)$ makes a clique. Figure 30 shows the class one clique on node 4 and 5 of the network in Figure 28. $\mathcal{Q}_1$ is stronger than $\mathcal{Q}_s$ since it is obtained by augmenting each star clique $E_{in}(u) \cup E_{out}(u)$ with $E(pred(u) \rightarrow v)$.

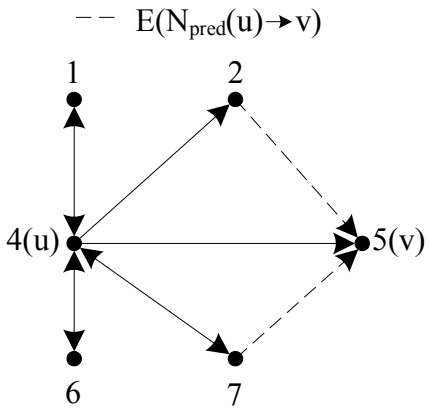


Fig. 30. Class one clique.

Second, let u, v and w be three nodes such that $uv \in E$ and $uw \in E$. The class two cliques is defined as follows:

$$\mathcal{Q}_2 = \{E_{out}(u) \cup E_{in}(v) \cup E(pred[v] \rightarrow w), \forall uv, uw \in E\}.$$

It is straightforward to verify that $E(pred[v] \rightarrow w)$ conflicts with $E_{out}(u)$ at w, and it also conflicts with $E_{in}(v)$ at v. We have already known $E_{out}(u) \cup E_{in}(v)$ is a bell clique, so $E_{out}(u) \cup E_{in}(v) \cup E(pred[v] \rightarrow w)$ is indeed a clique. Figure 31 shows the class two clique on node 4, 5 and 2 of the network in Figure 28. $\mathcal{Q}_2$ is stronger than $\mathcal{Q}_b$ since it is obtained by augmenting each bell clique $E_{in}(u) \cup E_{out}(v)$ with $E(pred[v] \rightarrow w)$.

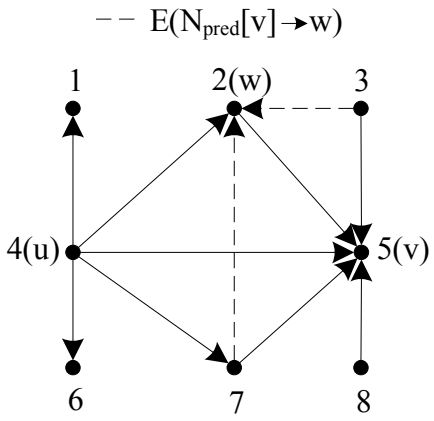


Fig. 31. Class two clique.

Finally, let u, v and w be three nodes such that $uv \in E$ and $wv \in E$. Consider class three cliques as follows:

$$\mathcal{Q}_3 = \{E_{out}(u) \cup E_{in}(v) \cup E(w \rightarrow succ[u]), \forall uv, wv \in E\}.$$

By similar analysis we can show that, each member of $\mathcal{Q}_3$ is indeed a clique too. As

an example, Figure 32 shows the class three clique on node 4, 5 and 7 of the network in Figure 28.

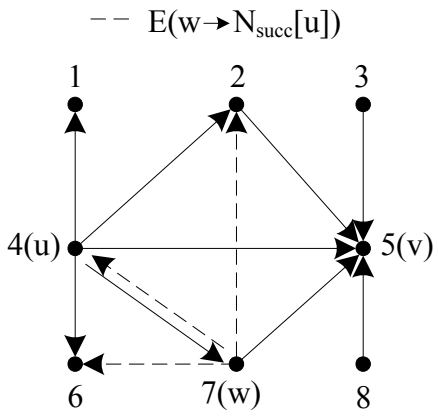


Fig. 32. Class three clique.

The asymmetricity in the form of $\mathcal{Q}_1$, $\mathcal{Q}_2$ and $\mathcal{Q}_3$ stems from the fact that the interference model is essentially a receiver oriented model, which precludes all the predecessors of the receiver but not the sender of a communication from transmitting. By the definition of the class one, two, and three cliques, there are at most $|E|$ cliques in $\mathcal{Q}_1$, and $|E|\cdot|V|$ cliques in $\mathcal{Q}_2$ and $\mathcal{Q}_3$ respectively. Thus, $\mathcal{Q}_1$, $\mathcal{Q}_2$ and $\mathcal{Q}_3$ can all be enumerated in polynomial time, and the corresponding upperbounds $\bar{\theta}_{sd}(\mathcal{Q}_1)$, $\bar{\theta}_{sd}(\mathcal{Q}_2)$, $\bar{\theta}_{sd}(\mathcal{Q}_3)$, and $\bar{\theta}_{sd}(\mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \mathcal{Q}_3)$ can be computed fairly efficiently. Furthermore, the simulation result to be presented shortly shows that these upperbounds are fairly tight in practice.

3. Simulation evaluation

In this section, we evaluate the clique based upperbounds on grid networks and random networks respectively. In our simulations, linear programs are solved by

Lp_solve[49], and the following simple recursive algorithm is used to compute the independence number α. The algorithm is based on the fact that $\alpha(F) = max\{\alpha(F\backslash\{e\}), \alpha(F\backslash I[e]) + 1\}$, where e is a link in F, $I[e] = I(e) \cup \{e\}$ and $I(e)$ is the set of links that e conflicts with.

1. if $|F| = 0$ return 0
2. pick a link $e \in F$
3. recursively compute $\alpha(F\backslash\{e\})$ and $\alpha(F\backslash I[e])$
4. return $max\{\alpha(F\backslash\{e\}), \alpha(F\backslash I[e]) + 1\}$

Grid networks

In this experiment, we create a 10x10 grid network. We assign to each link a random capacity with uniform distribution from 0 to 100. We randomly pick 10 pairs of nodes, and compute their $s - d$ throughputs. The upperbounds $\bar{\theta}_{sd}(\mathcal{Q}_1)$, $\bar{\theta}_{sd}(\mathcal{Q}_2)$, $\bar{\theta}_{sd}(\mathcal{Q}_3)$, and $\bar{\theta}_{sd}(\mathcal{Q}_1 \cup \mathcal{Q}_2 \cup \mathcal{Q}_3)$ are listed under column $\bar{\theta}_1$, $\bar{\theta}_2$, $\bar{\theta}_3$, and $\bar{\theta}_{123}$ in Table IV. Column $\underline{\theta}$ lists the lowerbounds computed by the maximal independent sets enumeration method in [50].

From the results we can see that, it always holds that $\bar{\theta}_1 > \bar{\theta}_2 = \bar{\theta}_3 = \bar{\theta}_{123}$, and the numbers in column $\bar{\theta}_{123}$ are very close to the numbers in column $\underline{\theta}$, which suggests that $\bar{\theta}_{123}$ is a practically good upperbound. In this regards, we computed $\bar{\theta}_{123}$ for all pairs of nodes in this network and compare them with $\underline{\theta}$. We found that more than 90% of the time, $\bar{\theta}_{123}$ is within $1.05\underline{\theta}$, and never goes above $1.25\underline{\theta}$.

Table IV. Grid network

	$\bar{\theta}_1$	$\bar{\theta}_2$	$\bar{\theta}_3$	$\bar{\theta}_{123}$	$\underline{\theta}$
1	60.04	53.06	53.06	53.06	51.33
2	53.41	51.62	51.62	51.62	51.41
3	86.00	86.00	86.00	86.00	86.00
4	73.79	64.71	64.71	64.71	56.62
5	63.84	60.14	60.14	60.14	51.89
6	62.79	54.70	54.70	54.70	54.70
7	60.62	47.70	47.70	47.70	41.17
8	44.13	41.57	41.57	41.57	34.67
9	33.20	31.36	31.36	31.36	31.07
10	33.20	31.36	31.36	31.36	31.28

Random networks

We create a set of 40 nodes on a 500x500 Euclidean plane. Each node is equipped with one radio with a random transmission range of mean 100 and variance 20. Each link in the network is assigned a random capacity with uniform distribution from 0 to 100. We randomly pick 10 pairs of nodes, and compute their $s - d$ throughputs. The results are listed in Table V.

From the results we can see that, again, $\bar{\theta}_{123}$ is a fairly close upperbound. After that, we computed $\bar{\theta}_{123}$ and $\underline{\theta}$ for all pairs of nodes in this network, and found that more than 90% of the time, $\bar{\theta}_{123}$ is within $1.35\underline{\theta}$, and never goes above $2\underline{\theta}$. Notice that, $\underline{\theta}$ is only a lowerbound of θ, thus, the gap between $\bar{\theta}_{123}$ and θ could be even smaller than that between $\bar{\theta}_{123}$ and $\underline{\theta}$.

Table V. Random network

	$\bar{\theta}_1$	$\bar{\theta}_2$	$\bar{\theta}_3$	$\bar{\theta}_{123}$	$\underline{\theta}$
1	22.99	23.10	22.72	22.71	17.66
2	8.17	7.53	7.53	7.53	7.53
3	41.18	39.90	40.81	39.90	27.91
4	20.67	16.47	16.47	16.47	16.47
5	8.17	8.17	8.17	8.17	8.17
6	84.00	84.00	84.00	84.00	84.00
7	96.00	96.00	96.00	96.00	96.00
8	67.38	61.90	55.77	54.57	44.96
9	51.58	38.85	37.82	35.42	32.00
10	8.17	7.53	7.53	7.53	7.53

E. Related work

The on-demand end-to-end bandwidth allocation problem that is considered in our discussion can be regarded as a variant of the QoS routing problem in wired networks. Compared with the QoS routing problem, our problem is more involved since it consists of not only a routing component, but also a scheduling component. Traditionally, QoS aware network layer design adopts minimum hop routing and its variations, such as shortest widest path routing [51], widest shortest path routing [52], and MPLS routing[53]. QoS routing via multiple paths using bandwidth reservation is studied in [54], [55], [56], and etc. These routing algorithms are not suitable for wireless networks because they are designed for networks where links are physically isolated. Wireless links, however, are not independent from each other, and allocating bandwidth on one link does affect the available bandwidth of others.

A variant of the end-to-end on-demand bandwidth allocation problem has been extensively studied in [57, 58, 59, 60, 61, 62, 63, 64]. The problem that they considered is as follows, given a path between a pair of source and sink, decide if a connection with a bandwidth requirement can be established along this path. Some slots of the links on that path may have already been allocated. The schedulability constraint is that nodes (on the path) cannot send and receive at the same time. [57] showed that this problem is NP-hard and [58, 59, 60, 61, 62] proposed various heuristic solutions. They differ from our work in that their focus is to satisfy the need of the current request, while ours is to optimize bandwidth utilization so that more future requests can be supported. Furthermore, a path is already given in their problem as the input. Therefore, only scheduling decision needs to be made in their problem formulation. While in our discussion, routing, i.e., finding a good path, is an indispensable part of the solution.

The problem of computing end-to-end throughput for multihop wireless networks was first studied in [65], where the system under study is assumed to be free of secondary interference. The author gave a polynomial time algorithm based on the famous ellipsoid method. The exact time complexity of their algorithm, however, is unknown. Following [65], [66] proposed a more practical algorithm that computes approximate throughput based on Shannon's coloring theorem. Both [65] and [66] neglect the existence of secondary interference. Jain et. al. showed this problem is NP-hard in general [50], and proposed a maximal independent set based lowerbounding and a clique based upperbounding technique. Their approach is general enough to be applied to a wide class of networks. However, it does not take full advantage of the special interference pattern of wireless networks. [67] first established a lowerbound that is within a constant factor of the throughput in wireless networks with secondary interference. Finally, [68, 69, 70] considered throughput estimation in

multi-radio multi-channel wireless networks by extending results for the single-radio single-channel case.

F. Summary

In this chapter, we studied two information transmission problems in wireless ad hoc networks.

We first considered a problem of end-to-end on-demand bandwidth allocation with the single transceiver schedulability constraint. The problem is a variant of the well-known QoS routing problem with an extra scheduling component. We proved that the problem is NP-hard and presented a 2-approximation routing and scheduling algorithm, MCRS. Our scheme exhibits better bandwidth efficiency than traditional methods in simulations. We also discussed the stateless version of the MCRS algorithm which is more amenable to distributed implementation. Our work can be extended along a variety of interesting directions. First, a most interesting question is whether we can find an algorithm whose approximation ratio is less than 2. Also, further empirical research is needed to study the performance of MCRS in practice. Second, our work is based on a deterministic TDMA network model. Although in general, deterministic access results in better performance, the overhead of synchronization and coordination may be significant. Hence, we plan to study bandwidth allocation in random access networks, especially those based on IEEE 802.11 medium access protocols. Third, we only studied QoS routing for unsplittable flows in our discussion. QoS routing for splittable flows is a rather interesting topic, where the data of a flow does not necessarily have to travel along a single path. Fourth, we focused on bandwidth efficient end-to-end unicasting in our discussion. Research on bandwidth efficient broadcasting and multicasting will also be of both theoretical

and practical interest. Fifth, there are many ways to define the optimality of a flow arrangement, and ours is only one of them. It would be interesting to adopt other notions of optimality, and design efficient algorithms correspondingly.

The second problem that we have studied is end-to-end throughput in multi-hop wireless ad hoc networks, which a variant of the classic maximum flow problem. We proposed a general linear program LP($\mathcal{F}$) based solution framework, where each choice of $\mathcal{F}$ corresponds to an upperbound of the end-to-end throughput. We identify several good choices of $\mathcal{F}$, the class one, two, and three cliques. We show by simulation that the upperbound based on these cliques is close to the actual throughput in practice.

CHAPTER IV

CONCLUSION

This dissertation studies ad hoc networks. The distinct characteristics of ad hoc networks include the lack of pre-existing infrastructure, the natural correlation between the network topology and geometry, limited communication and computation resources, node mobility, and etc. These characteristics created an enormous amount of new interesting and challenging problems, especially optimization problems. To effectively make use of the valuable ad hoc network resource, we studied a few optimization problems in information retrieval and transmission in this dissertation.

The study of ad hoc network information retrieval, in particular, the range search and object locating problem, is centered around the notion of distance sensitivity. Our main contribution is twofold. First, we developed a generic framework for distance sensitive services. This framework optimizes the storage cost for information retrieval schemes with an arbitrary retrieval cost function that can be specified by the system designer. We remark that, designing distance sensitive information retrieval systems with nonlinear retrieval cost functions is a largely unexplored area, and to our best knowledge, this dissertation is the first to consider such non-linear cases. Second, for growth lower bounded networks, we showed that an logarithmic storage cost is asymptotically optimal for any object locating schemes with linear retrieval cost. This result proves the optimality of a wide range of ad hoc network as well as P2P network proposals. We should point out that, despite the progress that we have made in this dissertation, there are many interesting questions that remain open, and the problem of distance sensitive information retrieval is still quite far from completely solved.

In the study of ad hoc network information transmission, our main concern is the various constraints for wireless transmissions to successfully occur, and the impact

of these constraints to effective information delivery. First, we studied a problem of end-to-end on-demand bandwidth allocation with the single transceiver schedulability constraint. We proved that the problem is NP-hard, proposed a 2-approximation routing and scheduling algorithm, MCRS, and showed by simulation that MCRS outperforms traditional methods. The second problem that we have studied is to compute the upperbound of the end-to-end throughput in wireless ad hoc networks. We investigated a general linear program based solution framework, and showed how to apply this general method under a specific radio interference model.

REFERENCES

[1] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, Mar 2000.

[2] C. E. Perkins, *Ad Hoc Networking*. Addison-Wesley, 2001.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications Magazine, IEEE*, vol. 40, no. 8, pp. 102–114, Aug 2002.

[4] R. Ramanathan and J. Redi, "A brief overview of ad hoc networks: challenges and directions," *Communications Magazine, IEEE*, vol. 40, no. 5, pp. 20–22, 2002.

[5] F. Kuhn, R. Wattenhofer, and A. Zollinger, "Worst-case optimal and average-case efficient geometric ad-hoc routing," in *MobiHoc '03: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 267–278, 2003.

[6] B. Karp and H. T. Kung, "Gpsr: greedy perimeter stateless routing for wireless networks," in *MobiCom '00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pp. 243–254, 2000.

[7] E. Kranakis, S. O. C. Science, H. Singh, and J. Urrutia, "Compass routing on geometric networks," in *Proc. 11th Canadian Conference on Computational Geometry*, pp. 51–54, 1999.

[8] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger, "Geometric ad-hoc routing: of theory and practice," in *PODC '03: Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing*, pp. 63–72, 2003.

[9] F. Zhang, H. Li, A. Jiang, J. Chen, and P. Luo, "Face tracing based geographic routing in nonplanar wireless networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pp. 2243–2251, May 2007.

[10] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic routing made practical," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, pp. 217–230, May 2005.

[11] B. Leong, B. Liskov, and R. Morris, "Geographic routing without planarization," in *NSDI'06: Proceedings of the 3rd Conference on 3rd Symposium on Networked Systems Design & Implementation*, pp. 25–25, 2006.

[12] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed. Springer-Verlag, 2000.

[13] I. Abraham, D. Dolev, and D. Malkhi, "Lls: a locality aware location service for mobile ad hoc networks," in *DIALM-POMC '04: Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*, pp. 75–84, 2004.

[14] R. Flury and R. Wattenhofer, "Mls: an efficient location service for mobile ad hoc networks," in *MobiHoc '06: Proceedings of the Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing*, pp. 226–237, 2006.

[15] M. Demirbas, A. Arora, T. Nolte, and N. Lynch, "Brief announcement: Stalk: a self-stabilizing hierarchical tracking service for sensor networks," in *PODC '04: Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, pp. 378–378, 2004.

[16] C. G. Plaxton, R. Rajaraman, and A. W. Richa, "Accessing nearby copies of replicated objects in a distributed environment," in *SPAA '97: Proceedings of*

the Ninth Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 311–320, 1997.

[17] I. Abraham, D. Malkhi, and O. Dobzinski, "Land: stretch (1 + e) locality-aware networks for dhts," in *SODA '04: Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 550–559, 2004.

[18] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, "Tapestry: An infrastructure for fault-tolerant wide-area location and," University of California at Berkeley, Berkeley, CA, Tech. Rep., 2001.

[19] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350, 2001.

[20] Mathworks, "matlab optimization toolbox." [Online]. Available: http://www.mathworks.com/products/optimization (accessed on Sep. 1, 2008).

[21] R. Sarkar, X. Zhu, and J. Gao, "Double rulings for information brokerage in sensor networks," in *MobiCom '06: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, pp. 286–297, 2006.

[22] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, "Data-centric storage in sensornets with ght, a geographic hash table," *Mob. Netw. Appl.*, vol. 8, no. 4, pp. 427–442, 2003.

[23] S. Funke and I. Rauf, "Information brokerage via location-free double rulings," in *ADHOC-NOW '07: the 6th International Conference on Ad-Hoc, Mobile, and Wireless Networks*, pp. 87–100, 2007.

[24] Q. Fang, J. Gao, and L. J. Guibas, "Landmark-based information storage and retrieval in sensor networks," in *INFOCOM '06: Proceedings of the 25th Conference on Computer Communications*, pp. 1–12, 2006.

[25] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM '01: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 149–160, 2001.

[26] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Transaction on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[27] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *SIGCOMM '01: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp. 161–172, 2001.

[28] M. F. Kaashoek and D. R. Karger, "Koorde: A simple degree-optimal distributed hash table," in *IPTPS '03: Proceedings of the 2nd International Workshop on Peer-to-Peer Systems*, pp. 98–107, 2003.

[29] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pp. 53–65, 2002.

[30] D. Malkhi, M. Naor, and D. Ratajczak, "Viceroy: a scalable and dynamic emulation of the butterfly," in *PODC '02: Proceedings of the Twenty-first Annual Symposium on Principles of Distributed Computing*, pp. 183–192, 2002.

[31] G. S. Manku, M. Naor, and U. Wieder, "Know the neighbor's neighbor: the power of lookahead in randomized p2p networks," in *STOC '04: Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, pp. 54–63, 2004.

[32] B. Awerbuch and D. Peleg, "Concurrent online tracking of mobile users," *SIGCOMM Comput. Commun. Rev.*, vol. 21, no. 4, pp. 221–233, 1991.

[33] D. Peleg and B. Awerbuch, "Online tracking of mobile users," *Journal of ACM*, vol. 42, no. 5, pp. 1021–1058, 1995.

[34] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris, "A scalable location service for geographic ad hoc routing," in *MobiCom '00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pp. 120–130, 2000.

[35] M. Thorup and U. Zwick, "Compact routing schemes," in *SPAA '01: Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 1–10, 2001.

[36] I. Abraham and D. Malkhi, "Compact routing on euclidian metrics," in *PODC '04: Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing*, pp. 141–149, 2004.

[37] D. Malkhi and I. Abraham, "Name independent routing for growth bounded networks," in *SPAA '05: Proceedings of the Seventeenth Annual ACM symposium on Parallelism in Algorithms and Architectures*, pp. 49–55, 2005.

[38] G. Konjevod, A. W. Richa, and D. Xia, "Optimal-stretch name-independent compact routing in doubling metrics," in *PODC '06: Proceedings of the Twenty-*

fifth Annual ACM Symposium on Principles of Distributed Computing, pp. 198–207, 2006.

[39] G. Konjevod, A. W. Richa, and D. Xia, "Optimal scale-free compact routing schemes in networks of low doubling dimension," in *SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 939–948, 2007.

[40] I. Abraham, C. Gavoille, A. V. Goldberg, and D. Malkhi, "Routing in networks with low doubling dimension," in *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, pp. 71–80, 2006.

[41] H. T.-H. Chan, A. Gupta, B. M. Maggs, and S. Zhou, "On hierarchical routing in doubling metrics," in *SODA '05: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 762–771, 2005.

[42] M. Arias, L. J. Cowen, K. A. Laing, R. Rajaraman, and O. Taka, "Compact routing with name independence," in *SPAA '03: Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, pp. 184–192, 2003.

[43] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, and M. Thorup, "Compact name-independent routing with minimum stretch," in *SPAA '04: Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 20–24, 2004.

[44] J. Joseph C. Liberti and T. S. Rappaport, *Smart Antennas for Wireless Communications*. Prentice Hall PTR, 1998.

[45] R. Ramanathan and R. Hain, "Topology control of multihop wireless networks

using transmit power adjustment," in *IEEE Conference on Computer Communications(Infocom)*, pp. 404–413, 2000.

[46] S. Ramanathan and E. L. Lloyd, "Scheduling algorithms for multihop radio networks," *IEEE/ACM Transaction on Networking*, vol. 1, no. 2, pp. 166–177, 1993.

[47] S. Ramanathan, "A unified framework and algorithm for channel assignment in wireless networks," *Wireless Networks*, vol. 5, pp. 81–94, 1999.

[48] S. O. Krumke, M. V. Marathe, and S. S. Ravi, "Models and approximation algorithms for channel assignment in radio networks," *Wireless Networks*, vol. 7, no. 6, pp. 575–584, 2001.

[49] "lp_solve." [Online]. Available: http://groups.yahoo.com/group/lp_solve (accessed on Sep. 1, 2008).

[50] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, "Impact of interference on multi-hop wireless network performance," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking(Mobicom)*, pp. 66–80, 2003.

[51] Z. Wang and J. Crowcroft, "Qos routing for supporting resource reservation," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, 1996.

[52] R. A. Guerin, A. Orda, and D. Williams, "Qos routing mechanisms and ospf extensions," in *IEEE Global Telecommunications Conference(Globecom)*, pp. 3–8, Phoenix, AZ, 1997.

[53] M. Kodialam and T. Lakshman, "Minimum interference routing with applications to mpls traffic engineering," in *INFOCOM 2000. Proceedings of the Nine-*

teenth Annual Joint Conference of the IEEE Computer and Communications Societies, pp. 884–893, 2000.

[54] N. Rao and S. Batsell, "Qos routing via multiple paths using bandwidth reservation," *INFOCOM '98. Proceedings of the Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 1, pp. 11–18, 1998.

[55] B. Wang and J. Hou, "Multicast routing and its qos extension: problems, algorithms, and protocols," *Network, IEEE*, vol. 14, no. 1, pp. 22–36, 2000.

[56] X. Lin and N. B. Shroff, "An optimization-based approach for qos routing in high-bandwidth networks," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1348–1361, 2006.

[57] C. Ferguson, "Routing in a wireless mobile cdma radio environment," Ph.D. dissertation, Computer Science Department, University of California, Los Angeles, 1996.

[58] C. R. Lin and J. Liu, "Qos routing in ad hoc wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1426 – 1438, 1999.

[59] H. C. Lin and P. C. Fung, "Finding available bandwidth in multihop mobile wireless networks," in *IEEE Vehicular Technology Conference(VTC)*, pp. 912 – 916, 2000.

[60] C. R. Lin, "On-demand qos routing in multihop mobile networks," in *IEEE Conference on Computer Communications(Infocom)*, pp. 1735–1744, 2001.

[61] C. R. Lin, "Admission control in time-slotted multihop mobile networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 1974 – 1983, 2001.

[62] C. Zhu and M. S. Corson, "Qos routing for mobile ad hoc networks," in *IEEE Conference on Computer Communications(Infocom)*, pp. 958– 967, 2002.

[63] W.-H. Liao, Y.-C. Tseng, and K.-P. Shih, "A tdma-based bandwidth reservation protocol for qos routing in a wireless mobile ad hoc network," *IEEE International Conference on Communications*, vol. 5, pp. 3186–3190, 2002.

[64] Q. Xue and A. Ganz, "Ad hoc qos on-demand routing (aqor) in mobile ad hoc networks," *J. Parallel Distrib. Comput.*, vol. 63, no. 2, pp. 154–165, 2003.

[65] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *ACM/IEEE Transactions on Information Theory*, vol. 34, no. 5, pp. 910–917, 1988.

[66] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless networks: the joint routing and scheduling problem," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking(Mobicom)*, pp. 42–54, 2003.

[67] V. A. Kumar, M. V. Marathey, S. Parthasarathyz, and A. Srinivasan, "Algorithmic aspects of capacity in wireless networks," in *International Conference on Measurement and Modeling of Computer Systems(Sigmetrics)*, pp. 133 – 144, 2005.

[68] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless with orthogonal channels," *ACM/IEEE Transactions on Networking*, vol. 13, pp. 868–880, 2005.

[69] M. Alicherry, R. Bhatia, and L. E. Li, "Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks," in *Proceedings*

of the 11th Annual International Conference on Mobile Computing and Networking(Mobicom), pp. 58–72, 2005.

[70] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking(Mobicom)*, pp. 73–87, 2005.

VITA

Hong Lu received his B.E. in Electrical Engineering in 1999, and M.S. degree in Computer Science in 2002, from Southeast University, China. He entered the Computer Science program at Texas A&M University in September 2002, and received his Ph.D. in 2008. His research interests include networking algorithms and hardware/software system co-design. He can be reached at: Hong Lu, Department of Computer Science, Texas A&M University, College Station, TX 77843. His email address is luhong@tamu.edu.