Mobile Networks and Applications 9, 49–61, 2004

Capacity Assignment in Bluetooth Scatternets – Optimal and Heuristic Algorithms *

GIL ZUSSMAN and ADRIAN SEGALL
Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel

Abstract. Bluetooth enables portable electronic devices to communicate wirelessly via short-range ad-hoc networks. Initially Bluetooth will be used as a replacement for point-to-(multi)point cables. However, in due course, there will be a need for forming multihop ad-hoc networks over Bluetooth, referred to as scatternets. This paper investigates the capacity assignment problem in Bluetooth scatternets. The problem arises primarily from the special characteristics of the network and its solution requires new protocols. We formulate it as a problem of minimizing a convex function over a convex set contained in the matching polytope. We develop an optimal algorithm which is similar to the well-known flow deviation algorithm and that calls for solving a maximum-weight matching problem at each iteration. A centralized heuristic algorithm with a relatively low complexity is also developed. Then, since in an ad-hoc network there is no central authority that is responsible for network optimization, a distributed heuristic algorithm is proposed. Finally, numerical results are presented and it is shown that the heuristic algorithms usually converge to results that are relatively close to the optimal results.

Keywords: Bluetooth, scatternet, capacity assignment, scheduling, Personal Area Networks (PAN)

1. Introduction

Recently, much attention has been given to the research and development of Personal Area Networks (PAN). These networks are comprised of personal devices, such as cellular phones, PDAs and laptops, in close proximity to each other. Bluetooth is an emerging PAN technology, which enables portable devices to connect and communicate wirelessly via short-range ad-hoc networks [6,7,20]. The basic Bluetooth network topology (referred to as a *piconet*) is a collection of slave devices operating together with one master. A multihop ad-hoc network of piconets in which some of the devices are present in more than one piconet is referred to as a *scatternet*.

Since its announcement in 1998, the Bluetooth technology has attracted a vast amount of research. However, the issue of capacity assignment in Bluetooth scatternets has been rarely investigated. Moreover, most of the research regarding network protocols has been performed via simulation. In this paper we formulate an analytical model for the analysis of the capacity assignment problem and propose optimal and heuristic algorithms for its solution.

Capacity assignment in communication networks focuses on finding the best possible set of link capacities that satisfies the traffic requirements, while minimizing some performance measure (such as average delay). Most capacity assignment models deal mainly with static networks in which a cost is associated with each level of link capacity (see [4] for a review of models). The following discussion shows that there is a need to study the capacity assignment problem in Bluetooth scatternets in a different manner:

- In contrast with a wired and static network, in an ad-hoc network, there is no central authority responsible for network optimization and there is no monetary cost associated with each level of link capacity.
- The nature of the network allows frequent changes in the topology and requires frequent changes in the capacities assigned to every link.
- Unlike other ad-hoc network technologies in which all nodes within direct communication from each other share a common channel, in Bluetooth only a subgroup of nodes (piconet) shares a common channel and capacity has to be allocated to each link.
- There are unique constraints imposed by the MAC layer protocol (see section 2.1).

Although inter and intra-piconet scheduling as well as scatternet formation have received considerable attention (see section 2.2), the issue of capacity assignment in scatternets has not been investigated. Baatz et al. [1,2] have made an attempt to deal with the problem, but have found out that it is a complex issue.[1] A scatternet capacity assignment protocol has to determine the capacities that should be allocated in each piconet, such that the network performance will be optimized. We envision that in the future, capacity assignment protocols will start operating once the scatternet is formed and will determine link capacities that will be dynamically allocated by scheduling protocols. Thus, *capacity assignment protocols* are the missing link between scatternet formation and scatternet scheduling protocols, and are required in order to improve the utilization of the scatternet bandwidth. We also anticipate that the optimal solution of the *capacity assignment problem* will improve the evaluation of heuristic scatternet scheduling algorithms.

* This research was supported by a grant from the Ministry of Science, Israel.

[1] In [1] and [2] the term *piconet presence schedule* is used to refer to a notion similar to *capacity assignment*.

Currently, our major interest is in algorithms for quasi-static capacity assignment that minimizes the average delay in the scatternet. The analysis is based on a static model with stationary flows and unchanging topology. Needless to say, we are interested in distributed algorithms. However, we shall also focus on centralized algorithms, which are required in order to gain insight to the network performance and to evaluate the performance of the distributed algorithms.

To the best of our knowledge, the work presented in this paper is the first attempt to analytically study the capacity assignment problem in Bluetooth scatternets. In the sequel we formulate the scatternet capacity assignment problem as a minimization of a convex function over a convex set contained in the polytope of the well-known *matching problem* [21, p. 608] and show that different formulations apply to bipartite and nonbipartite scatternets. The methodology used by Gerla et al. [12,22] is used in order to develop an *optimal algorithm* which is similar to the *flow deviation algorithm* [4, p. 458]. The main difference between the algorithms is that at each iteration we solve a *maximum-weight matching problem* instead of a *shortest path problem*. Then, we introduce a *centralized heuristic algorithm* whose complexity is much lower than the complexity of the optimal algorithm and whose performance is often close to that of the optimal algorithm. A *distributed heuristic algorithm*, which is based on the centralized heuristic, is also introduced. Finally, numerical results are presented and it is shown that the heuristic algorithms usually converge to results that are very close to the optimal results.

This paper is organized as follows. Section 2 gives a brief introduction to Bluetooth technology and discusses related work. In section 3, we present the model and in section 4 we formulate the scatternet capacity assignment problem for bipartite and nonbipartite scatternets. An algorithm for obtaining the optimal solution of the problem is presented in section 5. In section 6, we develop distributed and centralized heuristic algorithms for bipartite scatternets and in section 7 we present numerical results. Section 8 summarizes the main results and discusses possible extensions.

2. Background

2.1. Bluetooth technology

Bluetooth utilizes a short-range radio link, which operates in the 2.4 GHz license free ISM band. Since the radio link is based on frequency-hop spread spectrum, multiple channels (frequency hopping sequences) can co-exist in the same wide band without interfering with each other. Two or more units sharing the same channel form a *piconet*, where one unit acts as a *master* controlling the communication in the piconet and the others act as *slaves*.

Bluetooth channels use a frequency-hop/time-division-duplex (FH/TDD) scheme. The channel is divided into 625 μsec intervals called *slots*. The master-to-slave transmission starts in even-numbered slots, while the slave-to-master transmission starts in odd-numbered slots. Masters and slaves are allowed to send 1, 3 or 5 slots *packets*, which are transmitted in consecutive slots. A slave is allowed to start transmission in a given slot if the master has addressed it in the preceding slot. Information can only be exchanged between a master and a slave, i.e., there is no direct communication between slaves. Packets can carry synchronous information (voice link) or asynchronous information (data link).[2]

Multiple piconets in the same geographic area form a *scatternet*. Since Bluetooth uses packet-based communication over slotted links, it is possible to interconnect different piconets in the same scatternet. Hence, a unit can participate in two or more piconets, on a time-sharing basis, and even change its role when moving from one piconet to another. We will refer to such a unit as a *bridge*. For example, a bridge can be a master in one piconet and a slave in another piconet. However, a unit cannot be a master in more than one piconet.

2.2. Related work

Due to the special characteristics of Bluetooth networks, many theoretical and practical questions regarding their performance have been raised (a review of issues requiring research can be found in [18]). Two main issues that are related to capacity assignment and which received relatively much attention are scheduling and scatternet forming.

In the Bluetooth specifications [6], the capacity allocation by the master to each link in its piconet is left open. The master schedules the traffic within a *piconet* by means of polling and determines how bandwidth capacity is to be distributed among the slaves. Numerous heuristic intra-piconet scheduling algorithms have been proposed and evaluated via simulation (e.g., [8–10,15] and references therein). Johansson et al. [18] presented an overall architecture for handling scheduling in a *scatternet* and a family of inter-piconet scheduling algorithms (algorithms for masters and bridges). Several inter-piconet scheduling algorithms have been proposed and evaluated (e.g., [1,2,14,16,17,23,25,27]). Recently, analytical results regarding the delay in piconets have been presented in [14] and [29].

According to the architecture presented in [18], inter-piconet scheduling algorithms should deal with capacity allocation requests from applications or forwarding functions. Thus, the solution of the capacity assignment problem is a desirable input to scheduling algorithms such as the ones discussed above.

As mentioned before, capacity assignment protocols are the missing link between scheduling and scatternet formation algorithms. Research regarding scatternet topology and scatternet formation protocols has been recently receiving increasing attention (e.g., [5,24,26] and references therein). We expect that the solution of the capacity assignment problem will enable the evaluation of different topologies and, therefore, will improve the design of scatternet formation algorithms.

[2] In this paper we concentrate on networks in which only data links are used.

3. Model and preliminaries

Consider the connected undirected scatternet graph $G = (N, L)$. N will denote the collection of *nodes* $\{1, 2, \ldots, n\}$. Each of the nodes could be a master, a slave, or a bridge. The *bi-directional link* connecting nodes i and j will be denoted by (i, j) and the collection of bi-directional links will be denoted by L. For each node i, denote by $Z(i)$ the collection of its neighbors. We denote by $L(U)$ $(U \subseteq N)$ the collection of links connecting nodes in U.

Usually, capacity assignment protocols deal with the allocation of capacity to directional links. However, due to the tight coupling of the uplink and downlink in Bluetooth piconets,[3] we concentrate on the total bi-directional link capacity. Hence, we assume that the average packet delay on a link is a function of the total link flow and of the total link capacity. An equivalent assumption is that the uplink and the downlink flows are equal (symmetrical flows).

Let F_{ij} be the average bi-directional flow on link (i, j) and let C_{ij} be the capacity of link (i, j) (the units of F and C are bits/second). We assume that the average bi-directional flow is positive on every link ($F_{ij} > 0 \, \forall (i, j) \in L$). We define f_{ij} as the ratio between F_{ij} and the maximal possible flow on a Bluetooth link when using a given type of packets.[4] We also define c_{ij} as the ratio between C_{ij} and the maximal possible capacity of a link. It is obvious that $0 < f_{ij} \leqslant 1$ and that $0 \leqslant c_{ij} \leqslant 1$. We shall refer to f_{ij} as the *flow on link* (i, j) and to c_{ij} as the *capacity of link* (i, j). Accordingly, $\bar{c}$ will denote the vector of the link capacities and will be referred to as the *capacity vector*.

The objective of the capacity assignment algorithms, described in this paper, is to minimize the average delay in the scatternet. We define D_{ij} as the total delay per unit time of all traffic passing through link (i, j), namely:

Definition 1. D_{ij} is the average delay per unit of the traffic multiplied by the amount of traffic per unit time transmitted over link (i, j).

We assume that D_{ij} is a function of the link capacity c_{ij} only. We should point out that the *optimal algorithm* requires no explicit knowledge of the function $D_{ij}(c_{ij})$. We shall need to assume only the following reasonable properties of the function $D_{ij}(\cdot)$.

Definition 2. $D_{ij}(\cdot)$ is defined such that all the following holds:

1. D_{ij} is a nonnegative continuous decreasing function of c_{ij} with continuous first and second derivatives.
2. D_{ij} is convex.
3. $\lim_{c_{ij} \to f_{ij}} D_{ij}(c_{ij}) = \infty$.
4. $D'_{ij}(c_{ij}) < 0$ for all c_{ij} where D'_{ij} is the derivative of D_{ij}.

We note that the properties presented in definition 2 conform to the analytic and simulation results regarding the delay in piconets presented in [14] and [29].

Using definition 1, we shall now define the total delay in the network.

Definition 3. The *total delay in the network per unit time* is denoted by D_T and is given by

$$D_T = \sum_{(i,j) \in L} D_{ij}(c_{ij}).$$

Since the total traffic in the network is independent of the capacity assignment procedure, we can minimize the average delay in the network by minimizing D_T. A capacity vector that achieves the minimal average delay will be denoted by $\bar{c}^*$.

In section 6 we will develop *heuristic algorithms* and use a delay function based on *Kleinrock's independence approximation* [19] which is described in the following definition.[5] We will employ the same approximation in section 7 in order to describe a few computational results regarding the optimal algorithm.

Definition 4 (Kleinrock's independence approximation). When neglecting the propagation and processing delay, $D_{ij}(c_{ij})$ is given by

$$D_{ij}(c_{ij}) = \begin{cases} \dfrac{f_{ij}}{c_{ij} - f_{ij}}, & c_{ij} > f_{ij}, \\ \infty, & c_{ij} \leqslant f_{ij}. \end{cases}$$

A *capacity assignment algorithm* has to determine what portion of the slots should be allocated to each master–slave link. On the other hand, a *scheduling algorithm* has to determine which master–slave links should use any given slot pair. Hence, we define a scheduling algorithm as follows.

Definition 5. A *scheduling algorithm* determines how each slot pair is allocated. It does not allow transmission on two adjacent links in the same slot pair.

The Bluetooth specifications [6] do not require synchronization of masters' clocks. Since the clocks are not synchronized a guard time is needed in the process of moving

[3] A slave is allowed to start transmission only after a master had addressed it in the preceding slot.

[4] For example, currently the maximal flow on a symmetrical link, when using five slots unprotected data packets (DH5), is 867.8 Kbits/second.

[5] Although analytic results regarding the delay are available for simple scheduling regimes and simple topologies (e.g., [29]), as far as we know, no analytic results are available for complex topologies and sophisticated scheduling regimes. Thus, despite the fact that the Kleinrock's independence approximation does not fully model the delay in a scatternet, it has been shown in the past to provide a relatively good estimation for the delay in networks involving Poisson stream arrivals. Therefore, it is used for the development of *heuristic* capacity assignment algorithms. We note that that the development of the *optimal* algorithm does not require this approximation.

Table 1
A list of abbreviations.

Abbreviation	Meaning	Notes
SCA	Scatternet Capacity Assignment	Problem
SCAB	Scatternet Capacity Assignment in Bipartite Graphs	Problem
SCD	Scatternet Capacity Deviation	Algorithm
HCSCA	Heuristic Centralized Scatternet Capacity Assignment	Algorithm
HDSCA	Heuristic Distributed Scatternet Capacity Assignment	Algorithm

a bridge from one piconet to another. Yet, in order to formulate a simple analytical model we assume that the *guard times are negligible*.

Finally, we note that table 1 includes a list of abbreviations, used throughout this paper.

4. Formulation of the problem

Scatternet graphs can be bipartite graphs or nonbipartite graphs [5] (a graph is called bipartite, if there is a partition of the nodes into two disjoint sets S and T such that each edge connects a node in S with a node in T [21, p. 50]). Any scatternet graph in which no master is allowed to be a bridge is necessarily bipartite. For example, the scatternet graph described in figure 1A is bipartite. If masters are also bridges, the scatternet may be bipartite (e.g., figure 1B) or nonbipartite (e.g., figure 1C). A piconet whose master is also a bridge is non-active when the master is away. Therefore, scatternets where masters are bridges may result in poor bandwidth utilization [5].

We note that a few examples of bipartite and nonbipartite scatternets as well as a discussion of their performance can be found in section 7.

In this section, we shall formulate the *capacity assignment problem* for bipartite and nonbipartite scatternets. We will show that the formulation for nonbipartite scatternets is more complex than the formulation for bipartite scatternets.

4.1. Bipartite scatternets

When a bipartite scatternet graph is given, the nodes can be partitioned into two sets S and T such that no two nodes in S or in T are adjacent. Accordingly, the problem of *scatternet capacity assignment in bipartite graphs* (SCAB) is formulated as follows.

Problem SCAB.

Given: Topology of a bipartite graph and flows f_{ij}.

Objective: Find capacities c_{ij} such that the average packet delay is minimized:

$$\min D_T = \min \sum_{(i,j)\in L} D_{ij}(c_{ij}) \tag{1}$$

Subject to:

$$c_{ij} > f_{ij} \quad \forall (i,j) \in L, \tag{2}$$

$$\sum_{j\in Z(i)} c_{ij} \leqslant 1 \quad \forall i \in S, \tag{3}$$

$$\sum_{j\in Z(i)} c_{ij} \leqslant 1 \quad \forall i \in T. \tag{4}$$

The first set of constraints (2) is obvious. Constraints (3) and (4) result from the TDD scheme and reflect the fact that the total capacity of the links connected to a node cannot exceed the maximal link capacity. Due to the assumption that the *guard times are negligible*, in (3) and (4) we neglect the time needed in the process of moving a bridge from one piconet to another. Notice that it is easy to see that the convex set defined by (2)–(4) is contained in the *bipartite matching* polytope [21].

The formulation of the problem is based on the assumption that the flow rates are given. Although this is not the situation in a real scatternet, we assume that the traffic statistics can be evaluated by higher layer protocols and can be used by the capacity assignment algorithms. Hence, algorithms such as the ones described in the paper, which depend on estimates of the traffic, are expected to provide insight into the development of good (although suboptimal) capacity allocation schemes.

4.2. Nonbipartite scatternets

We shall now show that a formulation similar to the formulation of problem SCAB is not valid for nonbipartite scatternets. A simple example of a nonbipartite scatternet, given in [1], is illustrated in figure 2A. Constraint (2) and the constraint

$$\sum_{j\in Z(i)} c_{ij} \leqslant 1 \quad \forall i \in N \tag{5}$$

are not sufficient in order for the capacity vector to be feasible in this example. The capacities described in figure 2A satisfy (2) and (5), but are not feasible because in any scheduling algorithm no two neighboring links can be used simultaneously. If links (1, 2) and (1, 3) are in use for distinct halves of the available time slots, there are no free slots in which link (2, 3) can be in use. Thus, if $c_{12} = 0.5$ and $c_{13} = 0.5$, there is no feasible way to assign any capacity to link (2, 3).

Baatz et al. [1,2] suggest that a methodology for finding a *feasible* (not necessarily efficient) capacity assignment[6] will be based on minimum coloring of a graph and indicate that:

[6] Baatz et al. [1] refer to *piconet presence schedule* instead of *capacity assignment.* A piconet presence schedule determines in which parts of its time a node is present in each piconet. It is very similar to link capacity assignment as it is described in this paper.

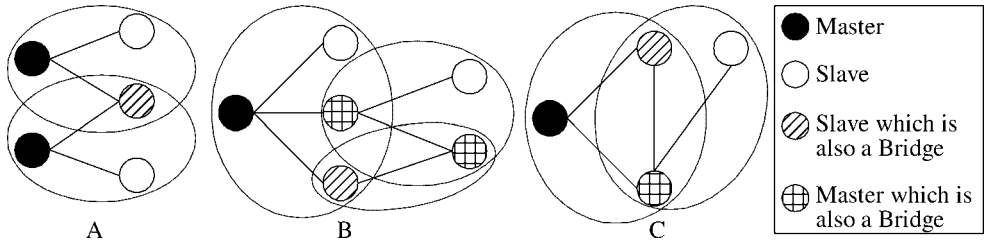


Figure 1. Scatternet graphs – a bipartite scatternet in which no master is also a bridge (A), a bipartite scatternet in which a master is also a bridge (B), and a nonbipartite scatternet (C).

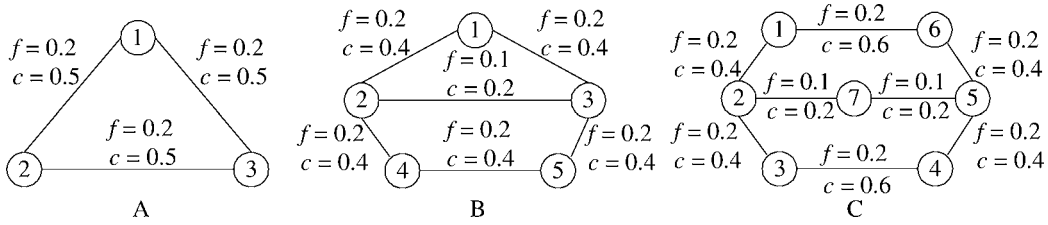


Figure 2. Examples of scatternets with capacity vectors which are not feasible.

"*the example gives an idea of how complex the determination of piconet presence schedules may get*". We propose a formulation of the problem that is based on the formulations of problem SCAB and the *matching problem* [21], and that allows obtaining an optimal capacity allocation.

The formulation of the capacity assignment problem for nonbipartite scatternets requires additional constraints to the constraints described in problem SCAB. For example, one could conclude that the capacity of the links composing the cycle described in figure 2A should not exceed 1. Since adjacent links cannot be active simultaneously, one could further conclude that the total capacity of links composing any odd cycle should not exceed: $(|\text{links}| - 1)/2$. Namely:

$$\sum_{(i,j)\in C} c_{ij} \leqslant \frac{|C| - 1}{2} \quad \forall C \subseteq L, C \text{ odd cycle.} \qquad (6)$$

However, in the examples given in figures 2B and 2C, although the capacities satisfy (6), they cannot be scheduled in any way. Thus, requirement (6) is still not sufficient to guarantee the feasibility of the capacity vector.

The set of links that are active in a slot must form a *matching* (a subset of links $X \subseteq L$ is said to be a matching if no two links in X are incident to the same node [21]). Edmonds [11] showed that the convex hull of the possible matching vectors in a graph is defined by a set of linear constraints (also known as the matching polytope). According to these constraints, in any odd set of nodes U the number of links that take part in a matching should not exceed $(|U| - 1)/2$. It follows from Edmonds' theorem and from the fact that the capacity of a link is the portion of the slots in which the link is active, that the capacity of links connecting nodes in any odd set of nodes U should not exceed $(|U| - 1)/2$.[7] The following lemma describes the constraints that result from this observation. We note that Hajek and Sasaki [13] have introduced a similar lemma regarding link scheduling in a certain kind of packet radio networks.

Lemma 1 (Edmonds [11]). The capacity vector must satisfy (2), (5), and the following constraints:

$$\sum_{(i,j)\in L(U)} c_{ij} \leqslant \frac{|U| - 1}{2} \quad \forall U \subseteq N, |U| \text{ odd}, |U| \geqslant 3. \quad (7)$$

The proof is based on Edmonds' theorem and can be found in [28].

The *scatternet capacity assignment* problem (SCA) can now be formulated as follows (for bipartite graphs it reduces to problem SCAB).

Problem SCA.

Given: Topology and flows f_{ij}.

Objective: Find capacities c_{ij} such that the average packet delay is minimized: (1)

Subject to: (2), (5) and (7).

The constraints (2), (5) and (7) form a *convex set* which is included in the matching polytope corresponding to the scatternet graph (for bipartite scatternets these constraints reduce to constraints (2)–(4) described in problem SCAB). This set consists of all the *feasible capacity vectors* $\bar{c}$. We shall now show that a feasible capacity vector has a corresponding scheduling algorithm. Namely, that it is possible to determine which links are used in each slot pair such that no two adjacent links are active in the same slot pair and the capacity used by each link is as defined by the capacity vector $\bar{c}$.[8] Since the vertices of the matching polytope defined by (5) and (7) are

[7] A similar observation has been recently independently made by Tassiulas and Sarkar [25] who have considered the problem of max–min fair scheduling in scatternets.

[8] Due to the TDD characteristics, each link should be active for at least a slot pair.

(0, 1) vectors, finding a scheduling algorithm is equivalent to finding the vertices whose convex combination provides $\bar{c}$. It can be found by the algorithm described in [13, section 3-C], which has $O(n^6)$ computation complexity.

5. Optimal algorithm

An optimal solution of the capacity assignment problem is required in order to gain some insight on the network's optimal design and in order to evaluate the performance of optimal and heuristic distributed algorithms. In this section we introduce a *centralized scatternet capacity assignment algorithm* for finding an optimal solution of problem SCA, defined in section 4.2.[9] The algorithm is based on the conditional gradient method also known as the Frank–Wolfe method [3, p. 215], which was used for the development of the flow deviation algorithm [4, p. 458]. Therefore, we refer to the algorithm as the *scatternet capacity deviation* (SCD) algorithm. Gerla et al. [12,22] have used the Frank–Wolfe method in order to develop bandwidth allocation algorithms for ATM networks. Following their approach, we shall now describe the optimality conditions and the algorithm.

Since the objective of problem SCA is to minimize a convex function (D_T) over a convex set ((2), (5) and (7)), any local minimum is a global minimum. Thus, necessary and sufficient conditions for the capacity vector $\bar{c}^*$ to be a global minimum are formulated as follows (the following proposition is derived from a well-known theorem [3, p. 194] and its proof is omitted).

Proposition 1. The capacity vector $\bar{c}^*$ minimizes the average delay for problem SCA, if and only if:

- $\bar{c}^*$ satisfies constraints (2), (5) and (7) of problem SCA.
- There are no feasible directions of descent at $\bar{c}^*$; i.e., there does not exist $\bar{c}$ such that:[10]

$$\nabla D_T\left(\bar{c}^*\right)\left(\bar{c}-\bar{c}^*\right) < 0, \tag{8}$$

$$\sum_{j\in Z(i)} c_{ij} \leqslant 1 \quad \forall i \in N, \tag{9}$$

$$\sum_{(i,j)\in L(U)} c_{ij} \leqslant \frac{|U|-1}{2} \quad \forall U \subseteq N, |U| \text{ odd}, |U| \geqslant 3. \tag{10}$$

Proposition 1 suggests a steepest descent algorithm in which we can find a feasible direction of descent $\bar{c}$ at any feasible point $\bar{c}^K$ by solving the problem:

$$\min \nabla D_T\left(\bar{c}^K\right)\bar{c} \tag{11}$$

subject to (9), (10) and

$$c_{ij} \geqslant 0 \quad \forall (i,j) \in L. \tag{12}$$

[9] The algorithm for the solution of problem SCAB is similar (the changes are outlined below).

[10] $\nabla D_T(\bar{c}^*)$ is the gradient of D_T with respect to $\bar{c}$ evaluated at $\bar{c}^*$.

1 **set** $K = 0$
2 **find** the vector $\bar{c}^{\#}$ – the optimal solution of (9)–(12) (i.e., solve a *maximum-weight matching* problem)
3 **find** the value α^* that minimizes $D_T(\alpha\bar{c}^K + (1-\alpha)\bar{c}^{\#})$
4 **set** $\bar{c}^{K+1} = \alpha^*\bar{c}^K + (1-\alpha^*)\bar{c}^{\#}$
5 **if** $\nabla D_T(\bar{c}^K)(\bar{c}^K - \bar{c}^{\#}) \leqslant t$
6 **then** stop
7 **else** set $K = K + 1$ and go to 2

Figure 3. Algorithm SCD for obtaining the optimal solution to problem SCA.

Since the constraint set (10) may include exponentially many constraints, this problem cannot be easily solved using a linear programming algorithm. Yet, since $D'_{ij}(c_{ij}) < 0$ for all c_{ij} (according to definition 2.4), the formulation of the problem conforms to the formulation of the *maximum-weight matching* problem [21, p. 610], which has a polynomial-time algorithm ($O(n^3)$).

This result and proposition 1 are the basis for algorithm SCD, described in figure 3. The input to the algorithm is the topology, the flows f_{ij}, a feasible initial solution $\bar{c}^0$, and the tolerance t. The output is the optimal capacity vector $\bar{c}^*$.

We emphasize that unlike the flow deviation algorithm, in which a feasible direction is found in each iteration by solving a shortest path problem, in algorithm SCD there is a need to solve a maximum-weight matching problem at each iteration. In case the algorithm is applied to problem SCAB, there is a need to solve a *bipartite* maximum-weight matching problem.

6. Heuristic algorithms for bipartite scatternets

As mentioned before, scatternet topologies in which a master is also a bridge may result in poor bandwidth utilization [5]. Scatternets in which a master is not allowed to be a bridge are necessarily bipartite (see section 4). In this section we propose low complexity *centralized* and *distributed* heuristic algorithms for such bipartite scatternets. We shall refer to these algorithms as the *heuristic centralized/distributed scatternet capacity assignment* algorithms (algorithm HCSCA/HDSCA, respectively).

The distributed algorithm is required because in an ad-hoc network, such as Bluetooth scatternet, there is no central authority that is responsible for network optimization. On the other hand, the centralized algorithm can be used to obtain the initial solution for the optimal algorithm (algorithm SCD, presented in section 5). The main difference between the centralized and distributed algorithms is the order in which capacity is allocated to the links.

In this section, we shall present the algorithms and show that the centralized algorithm always converges to a feasible capacity vector. We note that in our experiments (see section 7), the results of the heuristic algorithms were very close to the optimal results. Moreover, the two algorithms always converged to the same capacity vector.

The algorithms are based on the assumption that the delay function, presented in definition 4, is given. Before proceeding, we define the *slack capacity* of a node as follows.

Definition 6. The *slack capacity of node i* is the maximal capacity which can be added to links connected to the node. It is denoted by s_i and is given by

$$s_i = 1 - \sum_{j \in Z(i)} c_{ij}.$$

In both algorithms, all link capacities are initially equal to the flows on the links ($c_{ij} = f_{ij}\ \forall (i,\ j) \in L$).[11] The algorithms select a node and allocate the slack capacity to some of the links connected to it. Then, another node is selected, capacity is allocated and so on. Before describing the process of *node selection* we shall describe the process of *capacity allocation* and define the notions of *fully* and *non-fully allocated node*.

Once a node k is selected, *the slack capacity of this node is allocated to those adjacent links, whose capacities have not yet been assigned*. The slack capacity is assigned according to the square root assignment [19, p. 20]:

$$c_{kj} = f_{kj} + \frac{s_k \sqrt{f_{kj}}}{\sum_{m:\ m \in Z(k), c_{km} = f_{km}} \sqrt{f_{km}}}$$
$$\forall j:\ j \in Z(k),\ c_{kj} = f_{kj}. \tag{13}$$

It can be shown that after capacity is assigned to a subgroup of the links connected to a node i (links whose capacities have not been assigned before), the delay derivatives $D'_{ij}(c_{ij})$ of all these links will be the same. Accordingly, we define the *delay derivative of a node* as follows.

Definition 7. The *delay derivative of node i* is the square root of the absolute value of the delay derivatives of the links connected to node i, whose capacities have not yet been assigned. Its value is computed as if node i has been selected as the node whose capacity has to be assigned and the capacities of these links have been assigned according to (13). It is denoted by d_i and is given by

$$d_i = \frac{\sum_{m:\ m \in Z(i), c_{im} = f_{im}} \sqrt{f_{im}}}{s_i}. \tag{14}$$

According to (13), capacity is allocated to a link only once. Hence, we define the notions of a *fully allocated node* and a *non-fully allocated node* as follows.

Definition 8. A *fully allocated node* is a node such that all its adjacent link capacities have been assigned.[12]

Definition 9. A *non-fully allocated node* is a node such that at least one of its adjacent link capacities has not been assigned.

[11] In the distributed algorithm, when a node (i) receives a message of the algorithm for the first time it sets $c_{ij} = f_{ij}\ \forall j \in Z(i)$.

[12] A fully allocated node does not necessarily utilize its full capacity.

1 **set** $c_{ij} = f_{ij}\ \forall (i, j) \in L$
2 **set** $k = \underset{i \in N \cap i \text{ non-fully allocated}}{\arg\max}\, d_i$
3 **set** $c_{kj} = f_{kj} + \sqrt{f_{kj}}/d_k\ \forall j:\ j \in Z(k),\ c_{kj} = f_{kj}$
4 **if** there exists $(i, j) \in L$ such that $c_{ij} = f_{ij}$
5 **then** go to 2
6 **else** stop

Figure 4. Algorithm HCSCA for obtaining a heuristic solution to problem SCAB.

6.1. Centralized heuristic (algorithm HCSCA)

There are various ways to decide upon the order of *node selection*. For example, nodes can be selected according to their slack capacity or according to their average slack capacity. Some of the selection methodologies require care in order to ensure that the obtained capacity vector is feasible (satisfies constraints (2)–(4) of problem SCAB). We propose a simple selection methodology that not only guarantees a feasible capacity vector at each step, but also usually results in a vector which is close to the optimal capacity vector. The selection methodology is based on the square root assignment and on the property of the delay derivatives described in definition 7.

Node k, whose link capacities are next to be assigned, is selected from the non-fully allocated nodes. The delay derivatives d_i of these nodes are computed and the node with the largest derivative is selected. Thus, *the capacities of links with high absolute value of delay derivative, whose delay is more sensitive to the value of capacity, are assigned first*.

Algorithm HCSCA, which is based on the above methodology, is described in figure 4. The input is the topology and the flows f_{ij}, and the output is a capacity vector $\bar{c}$. It can be seen that the complexity of the algorithm is $O(n^2)$, which is about the complexity of a single iteration in the optimal algorithm. Moreover, the following proposition shows that the capacity vector obtained by the algorithm is always feasible.

Proposition 2. Algorithm HCSCA results in an allocation $\bar{c}$ that satisfies constraints (2)–(4) of problem SCAB.

The proof appears in the appendix.

6.2. Distributed heuristic (algorithm HDSCA)

In the distributed algorithm a *token* is passed by the nodes and only the node that holds the token is allowed to allocate capacity. The algorithm is initiated by an arbitrary node that creates the token.[13] Once a node receives the token, it can either allocate its slack capacity[14] or decide to send the token to a neighbor. The assignment of slack capacity is the same as in the centralized algorithm (square root assignment). However, the selection of the node which holds the token and the deci-

[13] At this stage of our work we assume that there is only one initiating node and that it initiates the protocol only once.

[14] Since bridges are unable to allocate capacity, we assume that if a bridge decides to allocate capacity, it informs its neighboring masters and they allocate the required capacities for it.

```
1  push the details of the node which sent the token to the parents stack
2  find the node with the largest d_k among the non-fully allocated neighbors
   and yourself*
3  if it is a neighbor
4     then send the token to this neighbor
5     else
6        allocate capacity according to (13)
7        update the neighbors
8        change the state to token transfer state
```

*According to definition 7, d_k is the value of the delay derivative of node k.

Figure 5. Algorithm HDSCA – the procedure executed by a node in the allocation state.

sion whether it should allocate capacity or transfer the token to a neighbor is different.

Each node keeps a stack, referred to as the *parents stack*, that contains the identities of neighbors from which it had previously received the token. Each node also maintains a list of non-fully allocated neighbors. We define two possible states for the node holding the token.

- *Allocation State*. A non-fully allocated node enters this state when it receives the token. At this time the node pushes the identity of the neighbor that sent it the token to the parents stack. The neighbor is referred to as one of the node's parents.[15] The node decides to either stay in this state and transfer the token to a neighbor or allocate capacity and then move to the token transfer state.
- *Token Transfer State.* A node enters this state after it allocates capacity or when it receives the token from a neighbor while being fully allocated. In this state, one of the non-fully allocated neighbors will receive the token. If all the neighbors are fully allocated, the token will be returned to the first neighbor in the stack and this neighbor will be popped from the stack. The protocol halts when all the neighbors are fully allocated and the stack is empty. The protocol always terminates at the initiating node.

Figure 5 presents the pseudocode of the procedure executed by a node in the allocation state. In the centralized algorithm a node allocates capacity if its d_k is the largest in the network. In the distributed algorithm a node allocates capacity if its d_k is larger than the d_k's of its neighbors. Thus, the order in which capacity is allocated to the links is not necessarily the same in the two algorithms. In our numerical experiments, the distributed algorithm (algorithm HDSCA) always converged to the same capacity vector as the centralized algorithm (algorithm HCSCA) regardless of the node which initiated the protocol. However, proving that this property holds for every scatternet and flow vector requires further research.

Figure 6 describes the pseudocode of the procedure executed by a node in the token transfer state. A node enters this state due to two possible events: capacity allocation by the node or receipt of the token from a neighbor that popped its details from the stack. In this state it can either send the token to its "best" neighbor or return it to one of its parents. Thus,

[15] Unlike other distributed protocols (such as Depth First Search), a node can have a few parents.

```
1  find the node with the largest d_k among the non-fully allocated neighbors
2  if such a node exists
3     then send the token to that neighbor
4     else if the stack is empty
5        then halt
6        else
7           pop the first node from the parents stack
8           send the token to that parent
```

Figure 6. Algorithm HDSCA – the procedure executed by a node in the token transfer state.

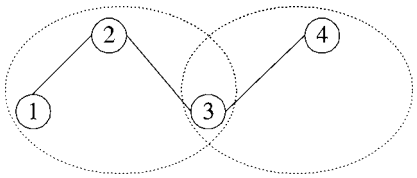


Figure 7. A simple scatternet.

the token either does not traverse a link or traverses it in both directions. It can be shown that since the token cannot be returned to a parent before all the neighbors are fully allocated, the algorithm cannot halt before all the link capacities have been allocated. Moreover, the protocol terminates at the initiating node, thereby providing an indication of the completion of the protocol.

In order to illustrate the need for a node to return the token to its parents, we shall describe a simple example of execution of the algorithm in the scatternet described in figure 7. Table 2 presents the operations performed during the execution, assuming that master 2 imitates the algorithm and that at initialization $d_3 > d_2$. Suppose that node 4 had another neighbor (referred to as node 5). If the algorithm stopped after node 2 allocates capacity (step 3 in table 2), link (4, 5) would have remained unallocated. However, since the token must be returned to a parent, it must be returned to bridge 3 at the end of step 3. This bridge should send it to its non-fully allocated neighbor – node 4, which would allocate the capacity of link (4, 5). Notice that in more complicated scatternet topologies a node may get the token from a few parents before it allocates capacity by itself.

Notice that the algorithm is based on token passing and thereby requires the implementation of mechanisms that deal

Table 2
The operations performed during the execution of Algorithm HDSCA (initiated by node 2) in the scatternet described in figure 7.

Step	Node	States	Operations
1	2	Allocation	Send token to node 3 ($d_3 > d_2$)
2	3	Allocation and Token Transfer	Allocate capacity and send token to node 2
3	2	Allocation and Token Transfer	Allocate capacity and send token to node 3 (parent)
4	3	Token Transfer	Send token to node 2 (parent)
5	2	Token Transfer	Halt (the parents stack is empty)

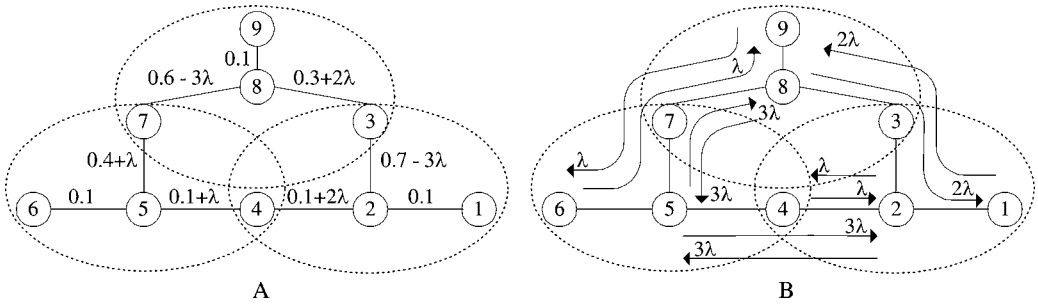


Figure 8. A bipartite scatternet with different values of flow (an arrow denotes flow along a path).

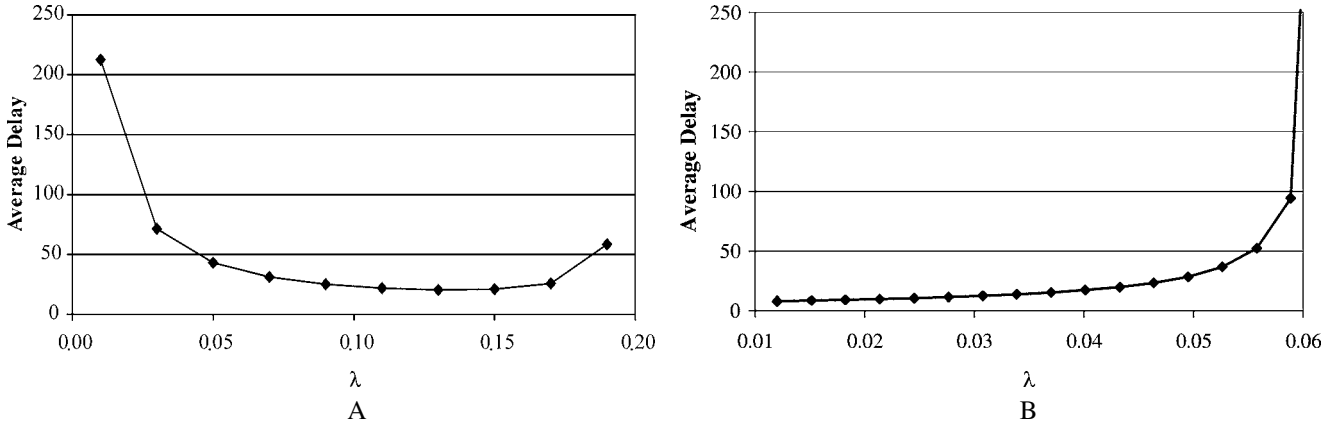


Figure 9. The optimal average delay in the scatternets presented in figures 8A and 8B.

with lost and duplicate tokens. These mechanisms will be implemented both in the link layer and as a part of the algorithm. Thus, in an operational version of the algorithm some of the nodes will be responsible for monitoring the network and dealing with situations in which a token is lost or duplicated.

7. Numerical results

The optimal algorithm (algorithm SCD, presented in section 5) and the heuristic algorithms (algorithm HCSCA and algorithm HDSCA, presented in section 6) were implemented[16] and tested on several representative cases. In all these cases, the results of the heuristic algorithms were very close to the optimal results and the centralized and distributed heuristic algorithms converged to the same capacity vector. Although our analysis is based on a static model with stationary flows, we have also evaluated the distributed heuristic in dynamic scenarios in which the flow rates change during the execution of the algorithm. It was found that the distributed heuristic usually converges to results close to the results in static scenarios. Regarding nonbipartite scatternets, it was found that in many cases the available capacity is utilized inefficiently relatively to its utilization in bipartite scatternets. In this section, we briefly describe the numerical results obtained for a few scatternets and demonstrate these findings.

[16] For the implementation of algorithm SCD, we used the delay function presented in definition 4.

7.1. Bipartite scatternets

Figures 8A and 8B illustrate a bipartite scatternet with different flow values. The scatternet topology is based on the topology presented in [24, figure 4] (every master has two bridge-slaves and one non-bridge slave). Figure 9 presents the optimal values of average delay (obtained by algorithm SCD) in the scatternets described in figure 8 for different values of λ. Figure 10 gives the ratio D_T/D_T^* where D_T is of the delay obtained by the centralized heuristic (algorithm HCSCA) and

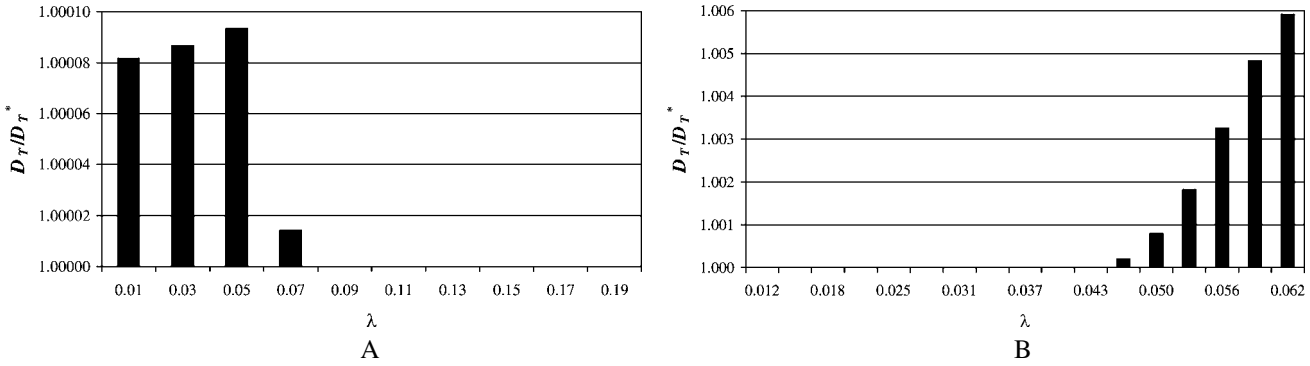


Figure 10. The ratio of the heuristic delay to the optimal delay (D_T/D_T^*) in the scatternets presented in figures 8A and 8B.

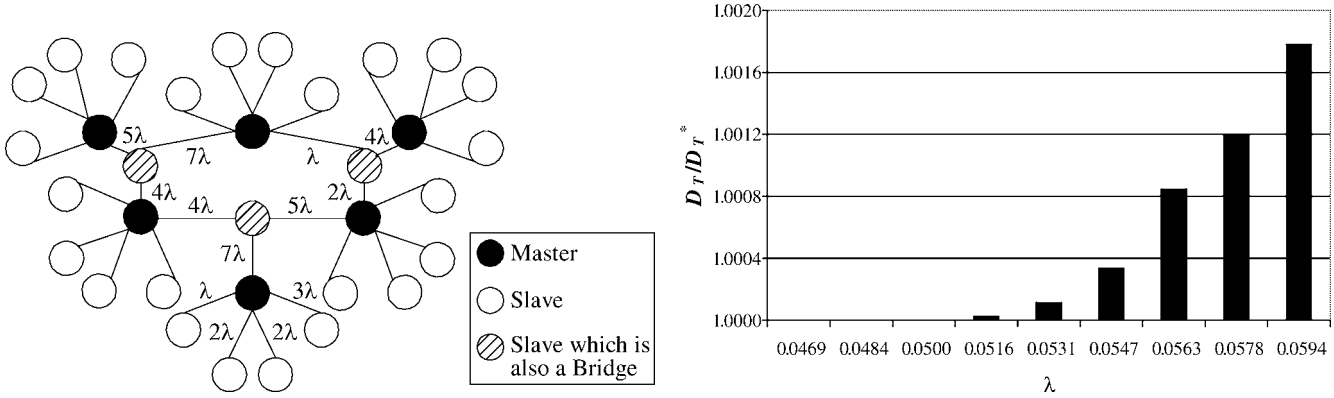


Figure 11. A bipartite scatternet (the flow values from the master to the non-bridge slaves in every piconet are identical to the values in the lowest piconet) and the ratio of the heuristic delay to the optimal delay (D_T/D_T^*).

D_T^* is the optimal delay. It can be seen that in all cases the two values are very close. For instance, in the worst case (the scatternet described in figure 8B with $\lambda = 0.062$) the heuristic delay is bigger than the optimal delay by 0.592%.

Figure 11 presents a more complex scatternet based on the topology described in [27, figure 1]. The figure also presents the ratio of the heuristic delay values to the optimal delay values, which as before is very close to 1.

There are cases in which algorithm SCD converges after a large number of iterations. However, the number of iterations is drastically reduced if the vector obtained by algorithm HCSCA is used as an initial solution for algorithm SCD. For example, table 3 includes the number of iterations required for obtaining the optimal solution with an arbitrary initial solution and with an initial solution computed by algorithm HCSCA.

We note that in our numerical experiments, the centralized heuristic (algorithm HCSCA) and the distributed heuristic (algorithm HDSCA) always converged to the same capacity vector although the two algorithms normally allocate capacity in different order.

As mentioned before, although the analysis is based on a static model with stationary flows, the distributed heuristic performed well in dynamic scenarios in which the flow rates vary during the execution of the algorithm. For example, we have simulated a scenario in which after a token transfer, the flow on every link $(i,\ j)$ is set to $f_{ij}(1+R)$ where R is a uniformly distributed random variable ($R \sim U(-r_{\max}, r_{\max})$). Then, the algorithm determines the next step according to the new flow values. The graphs in figure 12 present the distribution of the ratio D_T/D_T^* where D_T^* is the optimal delay and D_T is of the delay obtained by algorithm HDSCA when the flows were varied at every iteration.[17] It can be seen that when the flow values are within reasonable ranges, the effect on the delay is relatively insignificant.

7.2. Nonbipartite scatternets

Figure 13 illustrates three nonbipartite scatternets and the optimal capacities obtained by algorithm SCD for a given flow. In the scatternet described in figure 13B, no node utilizes its full capacity (every node is idle for at least 10% of its time slots) and in the scatternet described in figure 13C, only two nodes utilize their full capacity (nodes 2 and 5). Allocations in which most of the nodes are idle during some of the time slots are typical to nonbipartite scatternets. This is an outcome of constraint (7), described in lemma 1, which must be satisfied in nonbipartite scatternets.

According to a condition described in [13] and [25], if every node is allowed to utilize at most 2/3 of its capacity, constraint (7) will be satisfied. For example, in the scatternet described in figure 13A, if every link capacity is 1/3, (7) will be satisfied. However, since this is not a necessary condition,

[17] For every value of λ and $r_{\max}$ the scenario was simulated 100 times.

a node can utilize more than $2/3$ of its capacity and (7) will still be satisfied (for example, nodes 2 and 5 in the scatternet described in figure 13C utilize their full capacity). In view of the fact that bounding the capacity of a node to $2/3$ seems drastic, we wish to illustrate the effect of (7) on the capacity allocated in the scatternet. Thus, we shall compare the capacities allocated when condition (7) is enforced as opposed to when it is ignored.

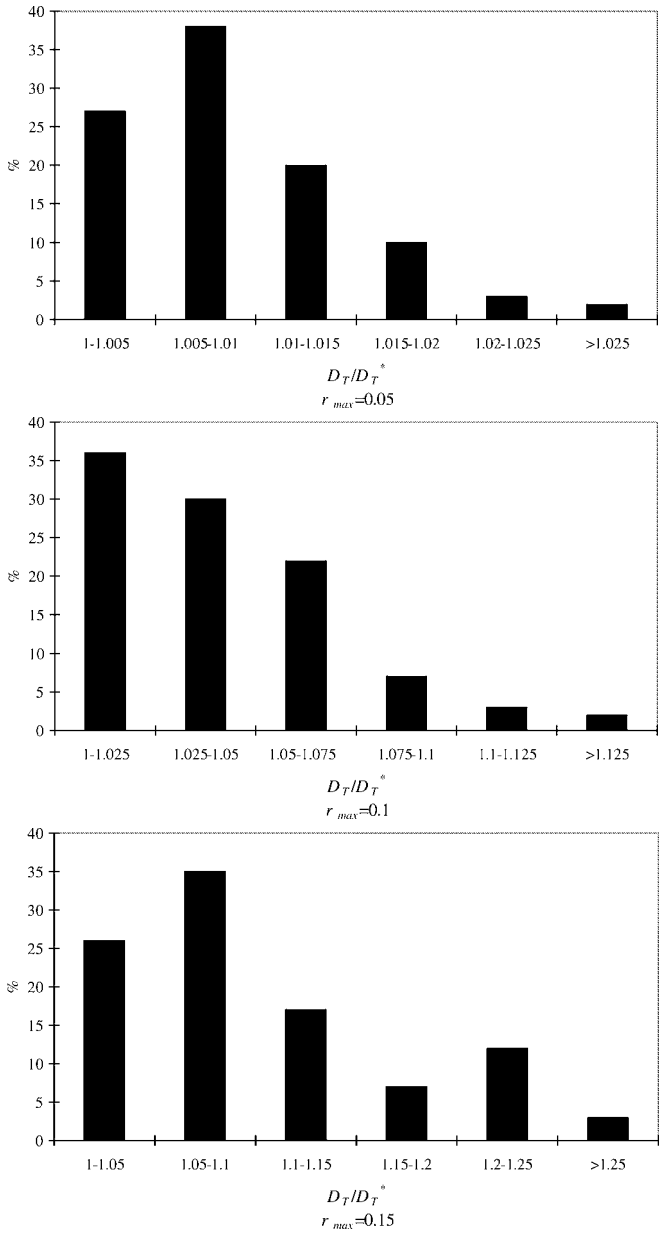


Figure 12. The distribution of the ratio of the heuristic delay, obtained for varying link flows, to the optimal delay (D_T/D_T^*) in the scatternet presented in figure 8B with $\lambda = 0.05$ and different values of $r_{\max}$.

For example, in the scatternet described in figure 13B, the average capacity allocated by a node is 0.8 and the total delay is 53.48. On the other hand, if (7) is ignored the average capacity is 0.95 and the total delay is 21.72. Thus, (7) causes a reduction of 15.8% in the average capacity allocated and an increase of 146% in the total delay. Notice that in this scatternet the node capacities cannot be restricted to $2/3$.

Another example of a nonbipartite scatternet is illustrated in figure 14. Figure 15 presents the average node capacities and the total delay when (7) holds and when it is ignored. When (7) holds the average node capacity is 0.8 (for all the values of λ, node 2 utilizes its full capacity) and when it is ignored the average node capacity is 0.867. Moreover, on average the delay is increased by 15.6% due to (7).

Although the constraint imposed by the nonbipartite topology does not seem to degrade the performance by $1/3$, it appears that a scatternet with a nonbipartite topology may utilize its resources in an inefficient manner. This conclusion, along with the fact that scatternet topologies in which a master is also a bridge may result in poor bandwidth utilization supports our focus on algorithms for bipartite scatternets.

8. Conclusions and future study

This paper presents an analytical study of the capacity assignment problem in Bluetooth scatternets. The problem has been formulated for bipartite and nonbipartite scatternets, using the properties of the matching polytope. Then, we have introduced a centralized algorithm for obtaining its optimal solution. A low complexity heuristic algorithm for the solution of the problem in bipartite scatternets, which obtains results that are relatively close to the optimal results, has also been

Table 3
The number of iterations required for obtaining the optimal solution in the scatternet described in figure 8B. The optimal solution was computed with an arbitrary initial solution and with an initial solution obtained by algorithm HCSCA (the required tolerance t was 0.005).

λ	Initial solution	
	Arbitrary	Obtained by HCSCA
0.03700	5.952	1
0.04325	42.340	36
0.04950	100	215
0.05575	392	352
0.06200	20.993	303

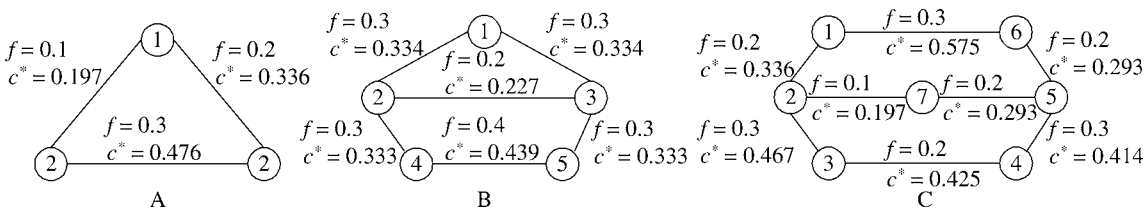


Figure 13. Nonbipartite scatternets and the optimal capacities c^* found by algorithm SCD for given values of flow f.

described. We have also developed a distributed heuristic that seems to obtain the same results as the centralized heuristic. Finally, we have presented a few numerical examples and discussed the performance of the heuristic algorithms in static and dynamic scenarios as well as the bandwidth utilization in nonbipartite scatternets.

The work presented here is the first approach towards an analysis of scatternet performance. Hence, there are still many open problems to deal with. For example, since distributed protocols are required for actual Bluetooth scatternets, future study will focus on developing *optimal* distributed protocols and improving the *heuristic* distributed protocol, presented in this paper. Furthermore, in this paper we have evaluated the performance of the distributed heuristic in scatternets with a static topology. In the future, we intend to investigate the performance of the distributed protocols in a dynamic topology maintained by a scatternet formation algorithm.

Finally, we note that a major future research direction is the development of capacity assignment protocols that will be able to deal with various quality-of-service requirements and to interact with scatternet formation, scheduling, and routing protocols.

Appendix

Proof of proposition 2

We shall introduce a lemma regarding algorithm HCSCA (described in figure 4) that is required in order to prove the proposition. The rest of the proof is by induction and it is omitted due to space constraints (it can be found in [28]).

Lemma 2. If in step 2 of algorithm HCSCA, node p is selected and in step 3 capacity is allocated to the link (p, q), then this capacity is smaller or equal to the capacity which would have been allocated to the link, in case node q had been selected in step 2.

Proof. Denote by $c_{pq}[p]$ the capacity of link (p, q) as it is allocated in step 3, following the selection of node p in step 2. Similarly, denote by $c_{pq}[q]$ the capacity of link (p, q) that would have been allocated in step 3, if node q had been selected in step 2.

In step 2, d_p and d_q are computed as if the nodes have been selected and the capacities have been assigned. Since node p is selected in step 2 and due to the selection procedure, $d_p \geqslant d_q$. According to (13) and definition 7, $D'_{pq}(c_{pq}[p]) = -d_p^2$. Therefore, $D'_{pq}(c_{pq}[p]) \leqslant D'_{pq}(c_{pq}[q])$.

According to definition 2.2, $D'_{ij}(c_{ij})$ is an increasing function of c_{ij} and therefore $c_{pq}[p] \leqslant c_{pq}[q]$. □

References

[1] S. Baatz, M. Frank, C. Kühl, P. Martini and C. Scholz, Adaptive scatternet support for Bluetooth using sniff mode, in: *Proc. of IEEE LCN'01* (November 2001) pp. 112–120.

[2] S. Baatz, M. Frank, C. Kühl, P. Martini and C. Scholz, Bluetooth scatternets: An enhanced adaptive scheduling scheme, in: *Proc. of IEEE INFOCOM'02* (June 2002) pp. 782–790.

[3] D.P. Bertsekas, *Nonlinear Programming* (Athena Scientific, MA, 1999).

[4] D.P. Bertsekas and R. Gallager, *Data Networks* (Prentice Hall International, Englewood Cliffs, NJ, 1992).

[5] P. Bhagwat and S.P. Rao, On the characterization of Bluetooth scatternet topologies, submitted for publication (August 2002), available at `http://www.winlab.rutgers.edu/~pravin/publications/papers/bt-top.ps`

[6] Bluetooth Special Interest Group, *Specification of the Bluetooth System – Version 1.1* (February 2001).

[7] J. Bray and C. Sturman, *Bluetooth 1.1 Connect without Cables* (Prentice Hall International, Englewood Cliffs, NJ, 2001) pp. 1990–1994.

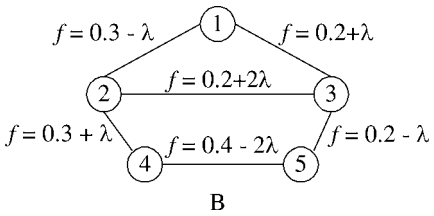
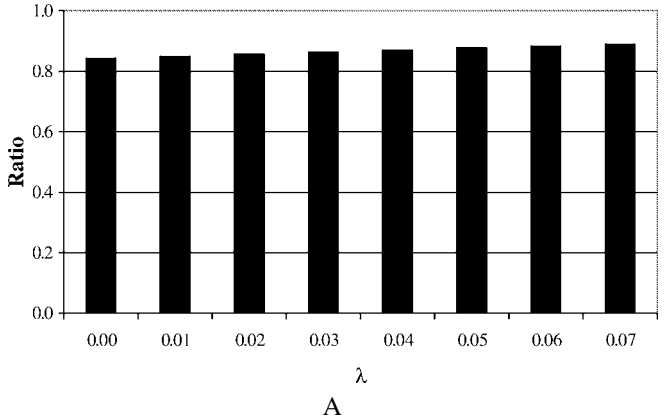


Figure 14. A nonbipartite scatternet.



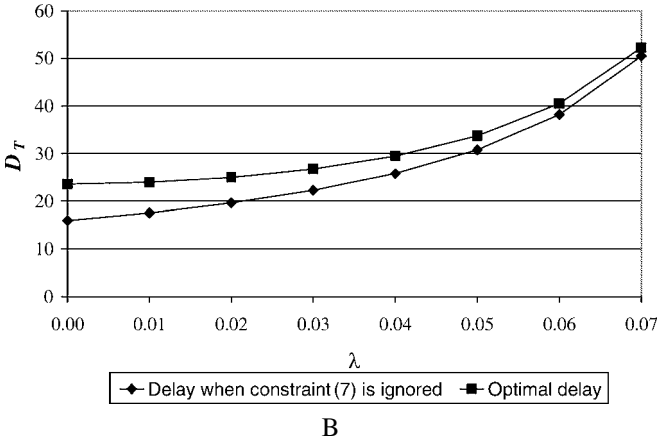


Figure 15. Results obtained for the scatternet presented in figure 14. The ratio of the average optimal node capacities to the average node capacities allocated when (7) is ignored (A) and a comparison of the optimal delay to the delay when (7) is ignored (B).

[8] R. Bruno, M. Conti and E. Gregori, Bluetooth: Architecture, protocols and scheduling algorithms, Cluster Computing 5 (April 2002) 117–131.
[9] A. Capone, M. Gerla and R. Kapoor, Efficient polling schemes for Bluetooth picocells, in: *Proc. of IEEE ICC'01* (June 2001) pp. 1990–1994.
[10] A. Das, A. Ghose, A. Razdan, H. Saran and R. Shorey, Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad-hoc network, in: *Proc. of IEEE INFOCOM'01* (April 2001) pp. 591–600.
[11] J. Edmonds, Maximum matching and a polyhedron with (0, 1) vertices, Journal of Research of the National Bureau of Standards 69B (1965) 125–130.
[12] M. Gerla, J.A.S. Monteiro and R.A. Pazos-Rangel, Topology design and bandwidth allocation in ATM nets, IEEE J. Selected Areas Commun. 7 (October 1989) 1253–1262.
[13] B. Hajek and G. Sasaki, Link scheduling in polynomial time, IEEE Trans. Inform. Theory 34 (September 1988) 910–917.
[14] L. Har-Shai, R. Kofman, G. Zussman and A. Segall, Inter-piconet scheduling in Bluetooth scatternets, in: *Proc. of OPNETWORK 2002* (August 2002).
[15] N. Johansson, U. Korner and P. Johansson, Performance evaluation of scheduling algorithms for Bluetooth, in: *Proc. of IFIP TC6 Internat. Conf. on Broadband Communications* (November 1999) pp. 139–150.
[16] N. Johansson, U. Korner and L. Tassiulas, A distributed scheduling algorithm for Bluetooth scatternet, in: *Proc. of ITC'17* (December 2001) pp. 61–72.
[17] P. Johansson, R. Kapoor, M. Kazantzidis and M. Gerla, Rendezvous scheduling in Bluetooth scatternets, in: *Proc. of IEEE ICC'02* (April 2002) pp. 318–324.
[18] P. Johansson, M. Kazantzidis, R. Kapoor and M. Gerla, Bluetooth: An enabler for personal area networking, IEEE Network 15 (September/October 2001) 28–37.
[19] L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay* (McGraw-Hill, New York, 1964).
[20] B.A. Miller and C. Bisdikian, *Bluetooth Revealed* (Prentice Hall International, Englewood Cliffs, NJ, 2000).
[21] G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization* (Wiley, New York, 1988).
[22] R.A. Pazos-Rangel and M. Gerla, Express pipe networks, in: *Proc. of Global Telecommunications Conf.* (1982) pp. B2.3.1–5.
[23] A. Racz, G. Miklos, F. Kubinszky and A. Valko, A pseudo-random coordinated scheduling algorithm for Bluetooth scatternets, in: *Proc. of ACM MOBIHOC'01* (October 2001) pp. 193–203.
[24] T. Salonidis, P. Bhagwat, L. Tassiulas and R. LaMaire, Distributed topology construction of Bluetooth personal area networks, in: *Proc. of IEEE INFOCOM'01* (April 2001) pp. 1577–1586.
[25] L. Tassiulas and S. Sarkar, Maxmin fair scheduling in wireless networks, in: *Proc. of IEEE INFOCOM'02* (June 2002) pp. 763–772.
[26] G.V. Zaruba, S. Basagni and I. Chlamtac, Bluetrees – scatternet formation to enable Bluetooth-based ad hoc networks, in: *Proc. of IEEE ICC'01* (June 2001) pp. 273–277.
[27] W. Zhang and G. Cao, A flexible scatternet-wide scheduling algorithm for Bluetooth networks, in: *Proc. of IEEE IPCCC'02* (April 2002) pp. 291–298.
[28] G. Zussman and A. Segall, Capacity assignment in Bluetooth scatternets – analysis and algorithms, CCIT Report, No. 355, Department of Electrical Engineering, Technion (October 2001), available at `http://www.comnet.technion.ac.il/segall/reports/CapacityBT.pdf`
[29] G. Zussman, U. Yechiali and A. Segall, Exact probabilistic analysis of the 1-limited scheduling algorithm for symmetrical Bluetooth piconets, in: *Proc. of IFIP-TC6 Personal Wireless Communications (PWC'03)*, eds. M. Conti et al., Lecture Notes in Computer Sience, Vol. 2775 (Springer, 2003) pp. 276–290.

Gil Zussman received the B.Sc. degree in industrial engineering and management and the B.A. degree in economics (both *summa cum laude*) from the Technion – Israel Institute of Technology in 1995. He received the M.Sc. degree (*summa cum laude*) in operations research from Tel-Aviv University in 1999. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering at the Technion. From 1995 to 1998, he served as an officer and an engineer in the Israel Defense Forces. His current research interests are in the area of ad hoc and sensor networks. In particular, he is interested in personal area networks, energy efficient routing, and medium access control protocols. Gil received the Knesset (Israeli Parliament) Award for distinguished students, the Best Student Paper Award at the IFIP-TC6 Networking 2002 Conference, and the IEEE Communications Magazine Best Paper Award at the OPNETWORK 2002 Conference.
E-mail: gilz@tx.technion.ac.il
WWW: http://www.comnet.technion.ac.il/~gilz

Adrian Segall received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion, Israel Institute of Technology in 1965 and 1971, respectively, and the Ph.D. degree in electrical engineering with a minor in statistics from Stanford University in 1973. After serving active duty in the Israel Defense Forces, he joined in 1968 the Scientific Department of Israel's Ministry of Defense. From 1973 to 1974 he was a Research Engineer at System Control Inc., Palo Alto, CA and a Lecturer at Stanford University. From 1974 to 1976 he was an Assistant Professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology. From 1987 to 1998 he was on the faculty of the Department of Computer Science at the Technion. He is presently Benjamin Professor of Computer-Communication Networks in the Department of Electrical Engineering, Technion, Israel Institute of Technology. From 1982 to 1984 he was on leave with the IBM T.J. Watson Research Center, Yorktown Heights, NY. He held visiting positions with IBM, AT&T and Lucent Bell Labs. His current research interests are in the area of optical networks, wireless, sensor and ad-hoc networks. Dr. Segall is an IEEE Fellow and has served in the past as Editor for Computer Communication Theory of the IEEE Transactions on Communications and Editor for the IEEE Information Theory Society Newsletter. He was selected as an IEEE delegate to the 1975 IEEE–USSR Information Theory Workshop, and is the recipient of the 1981 Miriam and Ray Klein Award for Outstanding Research and of the 1990 Taub Award in Computer Science. He is presently a Senior Editor for the IEEE Journal on Selected Areas in Communications.
E-mail: segall@ee.technion.ac.il
WWW: http://www.comnet.technion.ac.il/segall