IRWIN AND JOAN JACOBS
CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGIES

Capacity Assignment in Bluetooth Scatternets – Analysis and Algorithms

Gil Zussman and Adrian Segall

CCIT Report #355
October 2001

DEPARTMENT OF ELECTRICAL ENGINEERING
TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY, HAIFA 32000, ISRAEL

המרכז לטכנולוגיות תקשורת ומידע
הפקולטה להנדסת חשמל
הטכניון - מכון טכנולוגי לישראל, חיפה 32000, ישראל

Capacity Assignment in Bluetooth Scatternets – Analysis and Algorithms

Gil Zussman and Adrian Segall

Department of Electrical Engineering
Technion – Israel Institute of Technology
Haifa 32000, Israel
{gilz@tx, segall@ee}.technion.ac.il
http://www.comnet.technion.ac.il/segall

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 polytope contained in the matching polytope. Then, 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. Finally, a heuristic algorithm with a relatively low complexity is developed and numerical examples are presented.

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 [5],[6],[7],[14],[18]. Since its announcement in late spring 1998, the Bluetooth technology has attracted a vast amount of research. However, the issue of capacity assignment in Bluetooth networks has been rarely investigated. Moreover, most of the research regarding network protocols has been done 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.

Bluetooth utilizes a short-range radio link. 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. Although packets can carry synchronous information (voice link) or asynchronous information (data link), in this paper we concentrate on networks in which only data links are used.

Multiple piconets in the same area form a *scatternet*. Since Bluetooth uses packet-based communications over slotted links, it is possible to interconnect different piconets in the same scatternet. Hence, a unit can participate in different 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.

Initially Bluetooth piconets will be used as a replacement for point-to-(multi)point cables. However, in due course, there will be a need for multihop ad-hoc networks (scatternets). Due to the special characteristics of such networks, many theoretical and practical questions regarding the scatternet performance are raised. Nevertheless, only a few aspects of the scatternet performance have been studied. Two issues that received relatively much attention are: research regarding scatternet topology and development of efficient scatternet formation protocols (see [4],[16],[24] and references therein).

Much attention has also been given to scheduling algorithms for piconets and scatternets. 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 polling/scheduling algorithms for piconets have been proposed and evaluated via simulation (see for example [8],[9],[10],[13] and references therein). In [14] an overall architecture for handling scheduling in a scatternet has been presented and a family of inter-piconet scheduling algorithms (algorithms for masters and bridges) has been introduced. Inter-piconet scheduling algorithms have also been proposed in [1] and [21].

Although scatternet formation as well as piconet and scatternet scheduling have been studied, the issue of *capacity assignment* in Bluetooth scatternets has not been investigated. Moreover, Baatz et al. [1] who made an attempt to deal with it have indicated that it is a complex issue[1]. 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). We envision that in the future, scatternet capacity assignment protocols will start operating once the scatternet is formed and will determine link capacities that will be

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

dynamically allocated by scheduling protocols[1]. Thus, capacity assignment protocols are the missing link between scatternet formation and scatternet scheduling protocols. A correct use of such protocols will 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.

Most models of capacity assignment in communication networks deal mainly with static networks in which a cost is associated with each level of link capacity (see [3] and [22] for a review of models and algorithms). For example, in the problem presented by Bertsekas and Gallager [3, p. 439], the objective is to select link capacities so as to minimize the total cost, subject to the constraint that the average delay per packet should not exceed a given level.

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, there is no cost associated with each link and no budget constraint.
- The nature of the network allows frequent changes in the topology and requires frequent changes in the capacities assigned to every link.
- There are constraints imposed by the tight master-slave coupling and by the time-division-duplex (TDD) scheme.
- Unlike other ad-hoc networks 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.

A scatternet capacity assignment protocol has to determine the capacities that each master should allocate in its own piconet, such that the network performance will be optimized. Currently, our major interest is in algorithms for quasi-static capacity assignment that will minimize the average delay in the scatternet. The analysis is based on a static model with stationary flows and unchanging topology. To the best of our knowledge, the work presented in this paper is the first attempt to analytically analyze the capacity assignment problem in Bluetooth scatternets.

In this paper we focus on formulating the problem and developing centralized algorithms. The development of the distributed protocols is subject of further research.

In the sequel we show that the scatternet capacity assignment problem is more complex than it seems at first glance and that different formulations apply to bipartite and nonbipartite scatternets. We prove that the problem can be formulated as a minimization of a convex function over a polytope contained in the polytope of the well-known *matching problem* ([17],[19, p. 608]) and use this formulation in order to identify a few properties of the problem. The methodology used by Gerla et al.[12] and Pazos-Rangel and Gerla [20] is used in order to develop an *optimal scatternet capacity assignment algorithm* which is similar to the well-known *flow deviation algorithm* [11]. The main difference between the algorithms is that at each iteration

[1] This model conforms to the model presented in [14] in which the inter-piconet scheduling algorithm deals with capacity allocation requests from applications or forwarding functions.

there is a need to solve a *maximum-weight matching problem* instead of a *shortest path problem*.

We also introduce a *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. Finally, numerical examples regarding the optimal and heuristic algorithms are presented.

This paper is organized as follows. In Section 2, we present the model and in section 3 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 4. In Section 5, we develop a heuristic algorithm for obtaining the solution in a bipartite scatternet. Section 6 presents numerical examples and Section 7 summarizes the main results and discusses possible extensions.

2 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[1]. 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[2], 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 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). Without loss of generality, we assume that at every link the average bi-directional flow is positive ($F_{ij} > 0 \quad \forall (i,j) \in L$). 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[3]. We also define c_{ij} as the ratio between C_{ij} and the maximal possible capacity of a link[4]. It is obvious that $0 < f_{ij} \leq 1$ and that $0 \leq c_{ij} \leq 1$. In the sequel, f_{ij} will be referred to as the *flow on link* (i,j) and c_{ij} will be referred to as the *capacity of link* (i,j). Accordingly, $\overline{c}$ 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. Following Segall's formulation in [23],

[1] A bridge participates in different piconets, on a time-sharing basis. It can be a slave of a few masters or a master which is also a slave.

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

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

[4] The maximal capacity of a link is equal to the maximal flow on a link.

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 algorithms require 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}.*

Using Definition 1, we 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.

In Section 5 we will develop a heuristic algorithm and use a delay function based on *Kleinrock's independence approximation* [15] which is described in the following definition. We will employ the same approximation in Section 6 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*:

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

As we have mentioned, slot allocation is dynamic and it is managed by the masters. Accordingly, 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 that different masters' clocks will be synchronized. On the contrary, since the clocks not synchronized a guard time is needed in the process of moving a bridge from one piconet to another. Yet, in order to formulate a simple analytical model we assume the following.

Assumption 1. *The guard times are negligible.*

This assumption allows us to consider a scheduling algorithm for the whole scatternet (which master-slave links are active in each slot pair).

3 Formulation of the Problem

Scatternet graphs can be *bipartite graphs* or *nonbipartite graphs* [4] (see Fig. 1). For example, all the scatternets in which no master is allowed to be a bridge are bipartite[1]. In this section, we shall show that the formulation of the *capacity assignment problem* for nonbipartite scatternets is more complex than the formulation for bipartite scatternets. We will also identify a few properties of the capacity vector.

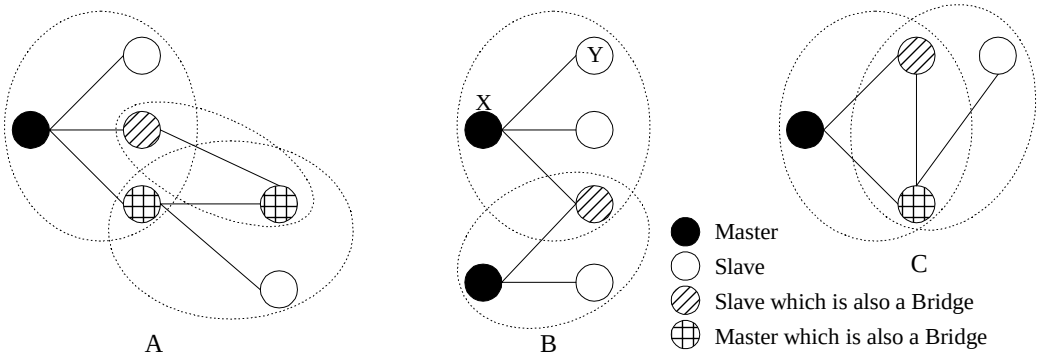

Fig. 1. Scatternet graphs – A bipartite scatternet (A), a bipartite scatternet in which no master is also a bridge (B), and a nonbipartite scatternet (C)

3.1 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) can be 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:

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

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

[1] Although it may be inefficient, according to Bluetooth specifications [6] a master can be a bridge.

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

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 capacity of a link. Due to Assumption 1, in (3) and (4) we neglect the guard time needed in the process of moving a bridge from one piconet to another. Notice that it is easy to see that the polytope defined by (2) - (4) is contained in the *bipartite matching* polytope [19].

3.2 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 Fig. 2-A. In this example, constraint (2) and the constraint:

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

are not sufficient in order for the capacity vector to be feasible. The capacities described in Fig. 2-A 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), i.e., there is no scheduling algorithm that can allocate the capacities described in the figure.

Baatz et al. [1] suggest that a methodology for finding a *feasible* (not necessarily efficient) capacity assignment[1] will be based on minimum coloring of a graph. They do not develop this methodology and indicate that: "*the example gives an idea of how complex the determination of piconet presence schedules may get*". In this paper, we propose a formulation of the problem that is based on the formulations of Problem SCAB and of the *matching problem* [19]. This formulation allows obtaining an optimal capacity allocation.

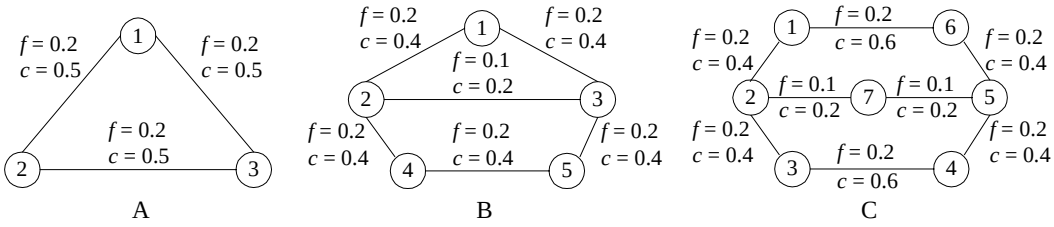

Fig. 2. Examples of scatternets with capacity vectors which are not feasible

[1] 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.

It is now obvious that 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 Fig. 2-A should not exceed 1. Moreover, one could further conclude that the total capacity of links composing any odd cycle should not exceed: (|links|-1)/2. Namely:

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

However, in the examples given in Fig. 2-B and Fig. 2-C, although the capacities satisfy (6), they cannot be scheduled in any way. Thus, in the following theorem we define a new set of constraints (9) such that the capacity of links connecting nodes in any odd set of nodes U will not exceed ($|U|$-1)/2. These constraints and the proof of the theorem are based on the properties of the matching problem.

Theorem 1. *The capacity vector must satisfy the following constraints*:

$$c_{ij} > f_{ij} \quad \forall (i,j) \in L \tag{7}$$

$$\sum_{j\in Z(i)} c_{ij} \le 1 \quad \forall i \in N \tag{8}$$

$$\sum_{(i,j)\in L(U)} c_{ij} \le \frac{|U|-1}{2} \quad \forall U \subseteq N,\ |U| odd,\ |U| \ge 3 \tag{9}$$

The proof appears in the appendix.

Notice that the polytope defined by (7) - (9) is included in the matching polytope corresponding to the scatternet graph. Notice also that constraint sets (7) - (9) include linearly dependent constraints. A similar theorem can be formulated without linearly dependent constraints by using the properties of two-connected factor-critical graphs [17]. However, since that formulation does not provide more insight regarding the problem and does not improve the optimal algorithm, we ignore it at this stage. We should also point out that for bipartite scatternets the constraints described in Theorem 1 reduce to constraints (2) - (4) described in Problem SCAB.

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: (7) - (9)

3.3 Properties of the Capacity Vector

Each of the constraint sets (2) - (4) and (7) - (9) forms a *convex set*. These sets consist of all the *feasible capacity vectors* ($\bar{c}$) for the corresponding problem (SCAB and SCA). A capacity vector that achieves the minimal delay will be denoted by $\bar{c}^*$.

Up to now we have formulated the problem and defined the conditions that must be satisfied by a capacity vector. Yet, we have not shown 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 at the same slot pair and the capacity used by each link is as defined by the capacity vector ($\overline{c}$). This result is shown by the following proposition. We note that the proof of the proposition and the transformation of a capacity vector to a scheduling algorithm are based on the fact that the vertices of the matching polytope are composed of (0,1) variables.

Proposition 1. *If a capacity vector $\overline{c}$ satisfies (7) - (9), there is a corresponding scheduling algorithm.*
The proof appears in the appendix.

Finally, it is easy to see that due to Definition 2.4, in an optimal allocation, some of the nodes must utilize their full capacity (i.e. for such nodes (3),(4) or (8) is satisfied with equality). Such nodes are connected to at least one node which has a single neighbor. On the other hand, in a scatternet consisting of more than two nodes, nodes that have a single neighbor cannot utilize their full capacity. For example, in an optimal allocation, master X in Fig. 1 must utilize its full capacity and slave Y cannot utilize its full capacity. These two properties are useful when trying to solve simple capacity assignment problems in scatternets composed of a few nodes and a few links, using the method of Lagrange multipliers. They are also useful in order to construct optimal capacity assignment algorithms.

4 Optimal Algorithm for Problems SCA and SCAB

In this section a *centralized scatternet capacity assignment algorithm* for finding an optimal solution of Problem SCA, defined in Section 3.2, is introduced[1]. The algorithm is based on the conditional gradient method also known as the Frank-Wolfe method [2, p. 215], which was used for the development of the flow deviation algorithm [11]. Therefore, we refer to the algorithm as the *scatternet capacity deviation* (SCD) algorithm.

Gerla et al. [12] and Pazos-Rangel and Gerla [20] 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 (7) - (9), any local minimum is a global minimum. Thus, necessary and sufficient conditions for the capacity vector $\overline{c}^*$ to be a global minimum are derived from the optimality conditions of convex functions over convex sets [2, p. 194] and are formulated as follows.

[1] The algorithm for the solution of Problem SCAB is similar (the changes are outlined in the sequel).

Proposition 2 *The capacity vector* $\overline{c}^*$ *minimizes the average delay for Problem SCA, if and only if*:

- $\overline{c}^*$ *satisfies constraints (7) - (9) of Problem SCA.*
- *There are no feasible directions of descent at* $\overline{c}^*$; *i.e. there does not exist* $\overline{c}$ *such that*[1]:

$$\nabla D_T(\overline{c}^*)(\overline{c}-\overline{c}^*) < 0 \tag{10}$$

$$\sum_{j\in Z(i)} c_{ij} \le 1 \quad \forall i \in N \tag{11}$$

$$\sum_{(i,j)\in L(U)} c_{ij} \le \frac{|U|-1}{2} \quad \forall U \subseteq N,\ |U| odd,\ |U| \ge 3 \tag{12}$$

As we have mentioned, the proposition is derived from a well-known theorem and, therefore, its proof is omitted. Notice that equation (10) means that D_T cannot be reduced by moving to $\overline{c}$, whereas (11) and (12) restrict $\overline{c}$ to the feasible region. Notice also that due to the property described in Definition 2.3 ($D_T \to \infty$ when $c_i \to f_i$), the first set of constraints of Problem SCA (7) is included in the objective function as a penalty function and is not needed in order to restrict $\overline{c}$ to the feasible region. It is easy to see that a similar optimality condition holds for Problem SCAB (explicitly, (11) and (12) should be replaced with (3) and (4)).

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

$$\min \nabla D_T(\overline{c}^K)\overline{c} \tag{13}$$

s.t.

$$\sum_{j\in Z(i)} c_{ij} \le 1 \quad \forall i \in N \tag{14}$$

$$\sum_{(i,j)\in L(U)} c_{ij} \le \frac{|U|-1}{2} \quad \forall U \subseteq N,\ |U| odd,\ |U| \ge 3 \tag{15}$$

$$c_{ij} \ge 0 \quad \forall (i,j) \in L \tag{16}$$

Since the constraint set (15) may include exponentially many constraints, this problem cannot be easily solved using a linear programming algorithm such as the simplex. 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 [19, p. 610], which has a polynomial-time algorithm ($O(n^3)$):

$$\max\left[-\nabla D_T(\overline{c}^K)\overline{c}\right] \tag{17}$$

s.t.

$$\sum_{j\in Z(i)} c_{ij} \le 1 \quad \forall i \in N \tag{18}$$

$$c_{ij} \in \{0,1\} \quad \forall (i,j) \in L \tag{19}$$

---

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

This result and the optimality conditions are the basis for Algorithm SCD, described in Fig. 3. The input to the algorithm is the scatternet topology, the flows (f_{ij}), a feasible initial solution ($\overline{c}^0$), and the tolerance (t). The output is the optimal capacity vector (within the desired tolerance) - $\overline{c}^*$.

1	Set $K = 0$
2	Find the vector $\overline{c}^{\#}$ - the optimal solution of (17) - (19) (i.e. solve a *maximum-weight matching* problem)
3	Find the value α^* that minimizes $D_T(\alpha\overline{c}^K + (1-\alpha)\overline{c}^{\#})$ (α^* may be obtained by any line search method [2, p. 723])
4	Set $\overline{c}^{K+1} = \alpha^*\overline{c}^K + (1-\alpha^*)\overline{c}^{\#}$
5	If $\nabla D_T(\overline{c}^K)(\overline{c}^K - \overline{c}^{\#}) \le t$ then stop
6	Else set $K = K+1$ and go to 2

Fig. 3. Algorithm SCD for obtaining an optimal solution to Problem SCA

We emphasize that unlike the flow deviation algorithm [11], in which at each iteration a feasible direction is found by solving a shortest path problem, in Algorithm SCD there is a need to solve a maximum-weight matching problem ($O(l^2 n)$) at each iteration. In case Algorithm SCD is applied to Problem SCAB, the constraints (14) - (16) reduce to the constraints of the *bipartite* maximum-weight matching problem.

5 Heuristic Algorithm for Problem SCAB

When considering *bipartite scatternets* (Problem SCAB), the initial solution for Algorithm SCD, introduced in the previous section, can be obtained using a low complexity *heuristic centralized scatternet capacity assignment* (HCSCA) algorithm, presented in this section. In our experiments (see Section 6), the results of the heuristic algorithm are very close to the optimal results.

The algorithm is based on the assumption that the delay function conforms to Kleinrock's independence approximation (i.e. the delay function presented in Definition 4). It assigns capacity to links connected to bridges and to masters which have at least two slaves. Accordingly, we define N' as follows:

Definition 6. *N' is a subgroup of N consisting of bridges and masters which have at least two slaves. Namely*:

$$N' = \left\{ i \;\middle|\; i \in N \cap \left| j \in Z(i) \right| > 1 \right\}$$

We also define the slack capacity of a node as follows:

Definition 7. *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*:

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

The algorithm selects a node from the nodes in N' and allocates the slack capacity to some of the links connected to that node. Then, it selects another node, allocates capacity and so on. The process of *capacity assignment* will be described first and then the process of *node selection* will be described.

Initially all the link capacities are equal to the flows on the links ($c_{ij} = f_{ij} \ \ \forall (i,j) \in L$). Once a node ($k$) is selected, *the slack capacity of this node is allocated to its links whose capacities have not yet been assigned* (links in which $c_{kj} = f_{kj}$). Thus, capacity is assigned to a link only once and the assignment affects the slack capacity of the selected node as well as the slack capacity of the neighboring nodes. The slack capacity is assigned to these links according to the square root assignment [15, p. 20]:

$$c_{kj} = f_{kj} + \frac{s_k \sqrt{f_{kj}}}{\sum_{m:\, m \in Z(k), c_{km} = f_{km}} \sqrt{f_{km}}} \quad \forall j:\ j \in Z(k), c_{kj} = f_{kj} \qquad (20)$$

According to [15] the square root assignment is the optimal capacity allocation. Therefore, if the links, which are connected to node k and whose capacities have not been assigned yet, were the only links in the network, the capacity would have been assigned optimally.

There are various ways to define process of *node selection*, i.e. to determine the order of capacity assignment to the nodes and, consequently, to the links connected to them. For example, nodes can be selected according to their slack capacity or their average slack capacity (slack per link). However, some of the possible selection methodologies require taking special measures in order to ensure that the obtained capacity vector is feasible (satisfies constraints (2) - (4) of Problem SCAB). We propose a selection methodology that not only guarantees a feasible capacity vector but also usually obtains a vector which is close to the optimal capacity vector. The selection methodology is based on a useful property of the square root assignment (20), described below.

Observe that when we use the delay function presented in Definition 4 and assign the capacity according to (20), the delay derivative of link (k,j) after the capacity of the links connected to k have been assigned is:

$$D_{kj}'(c_{kj}) = -\frac{f_{kj}}{(c_{kj} - f_{kj})^2} = -f_{kj} \left(\frac{s_k \sqrt{f_{kj}}}{\sum_{m:\, m \in Z(k), c_{km} = f_{km}} \sqrt{f_{km}}} \right)^{-2} = -\left(\frac{\sum_{m:\, m \in Z(k), c_{km} = f_{km}} \sqrt{f_{km}}}{s_k} \right)^2 \qquad (21)$$

It can be seen that $D_{kj}'(c_{kj})$ is a function of the flows and the capacities of the links connected to node k, and is independent of the specific value of c_{kj}. Therefore, when 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 equal. Accordingly, we can define the *delay derivative of a node* as follows.

Definition 8. *The* delay derivative of node *i is proportional to the absolute value of the delay derivatives of the links*[1] *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 (20). It is denoted by* d_i *and it is given by:*

$$d_i = \frac{\sum_{m:\, m\in Z(i),\, c_{im}=f_{im}} \sqrt{f_{im}}}{s_i} \tag{22}$$

We shall now define the notion of *fully allocated node*.

Definition 9. *A* fully allocated node *is a node that all its link capacities have been assigned*[2].

Node *k*, whose link capacities are going to be assigned, is selected from the nodes in N' which are not fully allocated. The delay derivatives (d_i's) of all these nodes are computed according to (22) and the node with the largest delay derivative is selected. Notice that since the delay derivatives are computed according to Definition 8, the delay derivative of a node is computed before the selection as if the node has been selected and its link capacities have been assigned.

Consequently, *the capacities of links with high absolute value of delay derivative, whose delay is more sensitive to the level of capacity, are assigned first.* Moreover, the following proposition shows that the capacity vector obtained by the algorithm is always feasible.

Proposition 3. *Algorithm HCSCA results in an allocation* $\{\bar{c}\}$ *that satisfies constraints (2) - (4) of Problem SCAB.*
The proof appears in the appendix.

Algorithm HCSCA, which is based on the above methodology, is described in Fig. 4. The input to the algorithm 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 an iteration in the optimal algorithm (Algorithm SCD).

1	Set $c_{ij} = f_{ij} \quad \forall (i,j) \in L$
2	Set $k = \underset{\substack{i \in N' \\ i \text{ not fully allocated}}}{\arg\max} \dfrac{\sum_{m:\, m\in Z(i),\, c_{im}=f_{im}} \sqrt{f_{im}}}{1 - \sum_{m\in Z(i)} c_{im}}$
3	Set $c_{kj} = f_{kj} + \dfrac{s_k \sqrt{f_{kj}}}{\sum_{m:\, m\in Z(k),\, c_{km}=f_{km}} \sqrt{f_{km}}} \quad \forall j:\ j \in Z(k), c_{kj} = f_{kj}$
4	If there exists $(i,j) \in L$ such that $c_{ij} = f_{ij}$ then go to 2
5	Else stop

Fig. 4. Algorithm HCSCA for obtaining a heuristic solution to Problem SCAB

[1] According to Definition 2.4 the delay derivative of a link is always negative.
[2] Notice that a fully allocated node does not necessarily utilize its full capacity.

6 Numerical Examples

The optimal algorithm (Algorithm SCD, presented in Section 4) and the heuristic algorithm (Algorithm HCSCA, presented in Section 5) were implemented[1] and tested on several representative cases. In this section we briefly describe the numerical results obtained for a few simple scatternets.

6.1 Bipartite Scatternets

Fig. 5 illustrates two bipartite scatternets whose capacity vectors were obtained by both algorithms[2]. The results obtained by the algorithms for different flow values are described in tables 1 and 2. We note that since the scatternet described in Fig. 5-A is relatively simple, the optimal capacities can also be easily computed using the method of Lagrange multipliers.

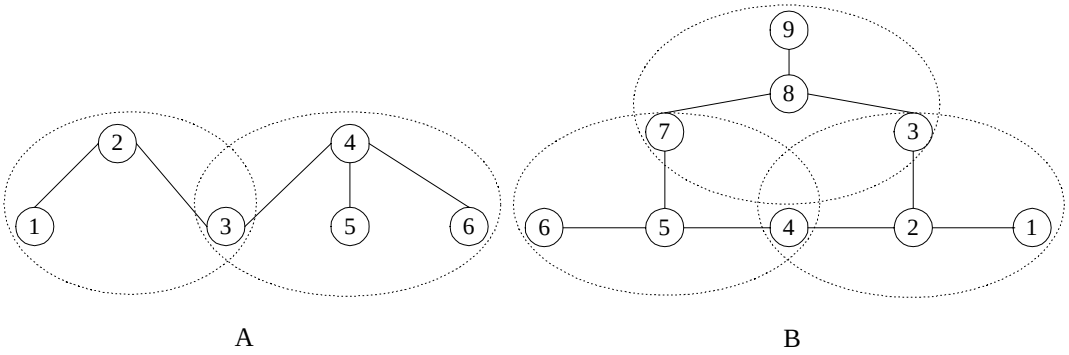

Fig. 5. Bipartite scatternets

Table 1. The results obtained by algorithms SCD and HCSCA for different flow values in the scatternet described in Fig 5-A

Link	(1,2)	(2,3)	(3,4)	(4,5)	(4,6)	D_T
Flows (f_{ij})	0.25	0.45	0.4	0.2	0.1	-
Optimal capacities (c_{ij}^*)	0.4713	0.5287	0.4713	0.3339	0.1948	15.01
Heuristic capacities (c_{ij})	0.4728	0.5272	0.4728	0.3331	0.1941	15.01
Flows (f_{ij})	0.2	0.4	0.2	0.01	0.1	-
Optimal capacities (c_{ij}^*)	0.3964	0.6036	0.3964	0.1283	0.4753	4.35
Heuristic capacities (c_{ij})	0.3657	0.6343	0.3657	0.1360	0.4983	4.45

[1] For the implementation of Algorithm SCD, we used a delay function based on Kleinrock's independence approximation (presented in Definition 4).

[2] An arbitrary vector was used as an initial solution to Algorithm SCD and the tolerance (t) was 0.005.

Table 2. The results obtained by algorithms SCD and HCSCA for different flow values in the scatternet described in Fig 5-B

Link	(1,2)	(2,3)	(2,4)	(4,5)	(5,6)	(5,7)	(3,8)	(7,8)	(8,9)	D_T
Flows (f_{ij})	0.02	0.4	0.5	0.45	0.05	0.45	0.3	0.4	0.05	-
Optimal capacities (c^*_{ij})	0.0291	0.4405	0.5304	0.4696	0.0576	0.4728	0.3975	0.5126	0.0898	85.68
Heuristic capacities (c_{ij})	0.0294	0.4420	0.5286	0.4714	0.0571	0.4714	0.3975	0.5126	0.0898	86.03
Flows (f_{ij})	0.1	0.2	0.5	0.4	0.1	0.3	0.1	0.2	0.1	-
Optimal capacities (c^*_{ij})	0.1608	0.2860	0.5531	0.4469	0.1560	0.3971	0.2757	0.4487	0.2757	28.73
Heuristic capacities (c_{ij})	0.1610	0.2862	0.5528	0.4472	0.1559	0.3969	0.2757	0.4485	0.2757	28.73

In general, there are cases in which Algorithm SCD converges after a large number of iterations. However, we note that after a few iterations, the algorithm usually converges to a vector which is close to the optimal vector. Moreover, it was found that there is usually a small difference between the optimal average delay and the average delay obtained by the heuristic algorithm. Accordingly, when the vector obtained by Algorithm HCSCA is used as an initial solution for Algorithm SCD, the algorithm converges relatively fast. Table 3 shows the number of iterations required for obtaining the optimal solution when the initial solution is an arbitrary vector and when it is computed by Algorithm HCSCA.

Table 3. The number of iterations required for obtaining the optimal solution by Algorithm SCD with an arbitrary initial solution and with an initial solution computed by Algorithm HCSCA

	Initial solution	
Scatternet and flows	Arbitrary vector	Obtained by HCSCA
Fig 5-A and *upper* part of Table 1	220	84
Fig 5-A and *lower* part of Table 1	30	2
Fig 5-B and *upper* part of Table 2	4,621	196
Fig 5-B and *lower* part of Table 2	1,194	118

6.2 Nonbipartite Scatternets

Fig. 6 illustrates two nonbipartite scatternets whose capacity vectors were obtained by Algorithm SCD[1] (the figure includes the given flows and the capacities found by the algorithm). Table 4 includes the values of total delay and the number of iterations required to obtain the optimal solutions.

It can be seen that in the scatternet described in Fig. 6-A, no node utilizes its full capacity (every node is idle for at least 10% of its time slots). In the scatternet described in Fig. 6-B, only two nodes utilize their full capacity (nodes 2 and 5). Moreover, in this scatternet significantly different capacities are allocated to links with identical flow requirement. Allocations in which most of the nodes are idle during some of the time slots are typical to nonbipartite scatternets. Thus, it seems

[1] An arbitrary vector was used as an initial solution and the tolerance (t) was 0.005.

that a scatternet with a nonbipartite topology may utilize its resources in an inefficient manner. However, this issue requires further research.

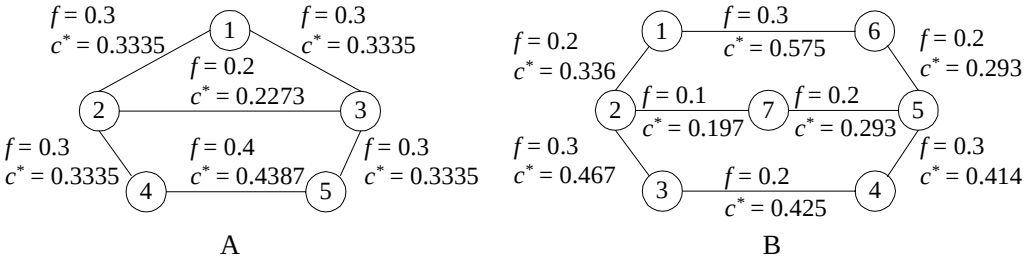

Fig. 6. Nonbipartite scatternets and the optimal capacities (c^*) found by Algorithm SCD for given values of flow (f)

Table 4. The total delay (D_T) and the number of iterations required for obtaining the results described in Fig. 6

	D_T	Number of iterations (K)
Scatternet described in Fig. 6-A	53.48	150
Scatternet described in Fig. 6-B	13.21	1,367

7 Conclusions and Future Study

This paper analytically analyzes the problem of capacity assignment 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 heuristic algorithm for the solution of the problem in bipartite scatternets, which has a relatively low complexity, has also been described. Finally, several numerical examples have been described.

The work presented here is the first approach towards an analytical analysis of the scatternet performance. Hence, there are still many open problems to deal with. For example, *distributed protocols* are required for actual Bluetooth scatternets and, therefore, future study will focus on developing optimal and heuristic distributed protocols. Moreover, according to Assumption 1, the guard times are negligible. This is of course not the situation in a real scatternet. Thus, in the future we intend to investigate the effect of more realistic assumptions on the formulation of the problem and its solution.

Furthermore, recall that Definition 2 describes a few assumptions regarding the properties of the delay function, and that we have assumed that the average packet delay on a *bi-directional* link is a function of the *total* link flow and capacity. An analytical model for the computation of bounds on the delay is required in order to evaluate these assumptions. In addition, it might enable developing efficient piconet scheduling algorithms.

Finally, we note that in the future, capacity assignment protocols will interact with protocols responsible for scatternet formation, scheduling, and routing. Thus, the main challenge is to develop a combined capacity assignment and inter-piconet scheduling protocol that will operate efficiently in the presence of other protocols.

Appendix

Proof of Theorem 1

According to Assumption 1, the guard times are negligible. Consequently, when a bridge moves from one scatternet to another it can start transmitting (in case it is a master) or receiving (in case it is a slave) immediately. Thus, according to Assumption 1 the slots at neighboring scatternets are synchronized and, therefore, the slots throughout the scatternet are synchronized.

Hence, we can define the following binary variables:

$$x_{ij}^{t} = \begin{cases} 1 & \text{if link } (i,\ j) \text{ is active at slot pair } t \\ 0 & \text{otherwise} \end{cases} \qquad \forall (i,\ j) \in L, \forall t \in \{1, 2, 3, ...\} \tag{23}$$

In slot pair t (an even slot and an odd slot) an active link (i,j) can be in one of the following states:

- The master transmits and the slave responds.
- The master or the slave transmits during the two slots a part of a 3 or 5 slots packet.

Therefore, an active link uses both slots in the slot pair. Thus, the definition of c_{ij} conforms to:

$$c_{ij} = \lim_{n \to \infty} \frac{\sum_{t=1}^{n} x_{ij}^{t}}{n} \tag{24}$$

Moreover, in any slot pair t no more than a single link connected to a node can be active and, therefore, the following must hold for every t:

$$\sum_{j \in Z(i)} x_{ij}^{t} \leq 1 \quad \forall i \in N \tag{25}$$

For a given t, constraint set (25) and the definition of the (0,1) variables in (23) describe a matching polytope, corresponding to the scatternet graph. This polytope can also be described by (25) and the following constraints [19]:

$$\sum_{(i,\ j) \in L(U)} x_{ij}^{t} \leq \frac{|U| - 1}{2} \quad \forall U \subseteq N,\ |U| odd,\ |U| \geq 3 \tag{26}$$

$$x_{ij}^{t} > 0 \quad \forall (i,\ j) \in L \tag{27}$$

Combining (25) and (26) with (24), and adding the flow constraints: $c_{ij} > f_{ij} \ \ \forall (i,j) \in L$ completes the proof. ■

Proof of Proposition 1

Constraint sets (7) - (9) define a polytope contained in the matching polytope corresponding to the scatternet graph. The matching polytope has only (0,1) vertices and, therefore, a vertex vector ($\overline{c}_k$) can be interpreted as a regime in which the links whose capacity is 0 are not active and the links whose capacity is 1 are active. Since a

vertex is a feasible solution of the matching problem, no two adjacent links can be active.

Since every feasible capacity vector $\overline{c}$ satisfies (7) - (9), it is a convex combination of the vertices of the matching polytope (denoted here by $\overline{c}_1, \overline{c}_2, \cdots, \overline{c}_m$). Namely:

$$\overline{c} = \alpha_1 \overline{c}_1 + \alpha_2 \overline{c}_2 + \cdots + \alpha_m \overline{c}_m$$

$$\sum_{k=1}^{m} \alpha_k = 1$$

$$\alpha_k \geq 0 \quad \forall k$$

In the proof of Theorem 1, we have mentioned that according to Assumption 1, the slots throughout the scatternet are synchronized. Therefore, every feasible capacity vector $\overline{c}$ can be interpreted as a scheduling regime, such that α_k of the slot pairs the links whose value in $\overline{c}_k$ equals 1 are active and the other links are not active. In this scheduling regime no two neighboring links are active at the same slot pair and the capacity used by each link is as defined by $\overline{c}$. ■

Proof of Proposition 3

We shall introduce a series of lemmas regarding Algorithm HCSCA (described in Fig. 4) that are required in order to prove the proposition. The first lemma establishes the basis for the proof and the other lemmas construct a proof by induction. Note that in this section we use the notion of *fully allocated node* (defined in Definition 9) and refer to a node which is not fully allocated as a *partially allocated node*. We also refer to an execution of steps 2-4 of the algorithm as *iteration*.

Lemma 1. *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: $d_p \geq d_q$. According to (21) and Definition 8: $D_{pq}'(c_{pq}[p]) = -d_p^2$. Therefore: $D_{pq}'(c_{pq}[p]) \leq D_{pq}'(c_{pq}[q])$.

According to Definition 2.2, $D_{ij}'(c_{ij})$ is an increasing function of c_{ij} and therefore: $c_{pq}[p] \leq c_{pq}[q]$ ■

Lemma 2. *If before an iteration,* $s_i > 0$ *for every partially allocated node i, then after the iteration,* $s_i > 0$ *for every partially allocated node i.*

Proof: In step 3, the capacities of the links connected to a certain partially allocated node (k) are computed. This step affects the slack capacity of node k and of the neighboring nodes connected to k by a link whose capacity has not yet been assigned.

Among these nodes, the only nodes that will still be partially allocated nodes after the iteration are neighboring nodes that have *more than one link* whose capacity has not yet been allocated.

Consider a node *p*, connected to node *k* by a link whose capacity has not yet been assigned, that has more than one link whose capacity has not been allocated (without loss of generality we assume that such a node exists). It follows from Lemma 1 that c_{kp} is smaller or equal to the capacity that would have been allocated to link (k,p), if node *p* had been selected in step 2. Due to (20), if node *p* had been selected in step 2, after the iteration the following would hold:

$$\sum_{i \in Z(p)} c_{pi} = 1$$

Therefore, when node *k* is selected in step 2, after the iteration the following holds:

$$\sum_{i \in Z(p)} c_{pi} < 1$$

and obviously: $s_p > 0$ also holds.

After the iteration, $s_i > 0$ for every partially allocated node *i* which is *not* a neighboring node of *k*.[1] We have shown that after the iteration, $s_i > 0$ for every partially allocated node *i* which is a neighboring node of *k*. Hence, after the iteration, $s_i > 0$ for every partially allocated node *i*. ■

Lemma 3. *Algorithm HCSCA results in an allocation* $\{\hat{c}\}$ *that satisfies* $c_{ij} > f_{ij} \quad \forall (i,j) \in L$.

Proof: Problem SCAB has a feasible solution only if after step 1: $s_i > 0 \quad \forall i \in N$. Therefore, according to Lemma 2, $s_i > 0$ for every partially allocated node *i* at the end of any iteration.

In step 3 of any iteration, the capacities of the links connected to a certain partially allocated node (*k*) are computed. In this step the capacity is allocated to links, connected to node *k*, whose capacities have not yet been assigned. Due to (20) and the fact that $s_k > 0$, the capacity is allocated such that $c_{kj} > f_{kj}$ for each of these links. It follows that when the algorithm halts: $c_{ij} > f_{ij} \quad \forall (i,j) \in L$. ■

Lemma 4. *Algorithm HCSCA results in an allocation* $\{\hat{c}\}$ *that satisfies:*

$$\sum_{j \in Z(i)} c_{ij} \le 1 \quad \forall i \in N \tag{28}$$

Proof: In step 3 of any iteration, the capacities of the links connected to a certain partially allocated node (*k*) are computed. Then, node *k* becomes a fully allocated node. Due to (20), after the iteration the following holds:

$$\sum_{i \in Z(k)} c_{ki} = 1$$

Other nodes that may become fully allocated are neighbors of node *k* connected to *k* by a link whose capacity has not been assigned. There could be two types of such a neighboring node *p*:

[1] The slack capacities of these nodes are not affected by the iteration.

– A node p such that $p \notin N'$ (node p is neither a relay nor a master with at least two slaves). It is obvious that for such a node the following holds:

$$\sum_{i \in Z(p)} c_{pi} \leq 1 \tag{29}$$

– A node p such that $p \in N'$ and the only link whose capacity has not been assigned until the iteration is (k,p). It follows from Lemma 1 that c_{kp} is smaller or equal to the capacity that would have been allocated to the link, if node p had been selected in step 2. Thus, for such a node (29) is satisfied.

Thus, if before the iteration:

$$\sum_{j \in Z(i)} c_{ij} \leq 1 \quad \forall i \in N\,,\ i \text{ fully allocated} \tag{30}$$

after the iteration, (30) is still satisfied.

Problem SCAB has a feasible solution only if after step 1 the following holds:

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

Thus, (30) holds after the first iteration and, consequently, it is satisfied at the end of any iteration. Since at the end of the last iteration all the nodes are fully allocated, at that stage (28) holds. ■

Proof of Proposition 3: The proposition follows from lemmas 3 and 4. ■

References

[1] S. Baatz, M. Frank, C. Kühl, P. Martini and C. Scholz, "Adaptive Scatternet Support for Bluetooth using Sniff Mode", *Proc. IEEE LCN'01*, Nov. 2001.

[2] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Massachusetts, 1999.

[3] D. P. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall Inc., New Jersey, 1992.

[4] P. Bhagwat and S. P. Rao, "On the Characterization of Bluetooth Scatternet Topologies", Submitted for publication, Oct. 2001.

[5] The Bluetooth Special Interest Group, Documentation available at URL http://www.bluetooth.com, Oct. 2001.

[6] Bluetooth Special Interest Group, *Specification of the Bluetooth System - Version 1.1*, Feb. 2001.

[7] J. Bray and C. Sturman, *Bluetooth connect without cables*, Prentice Hall, 2001.

[8] R. Bruno, M. Conti and E. Gregori, "Wireless Access to Internet via Bluetooth: Performance Evaluation of the EDC Scheduling Algorithm", *Proc. ACM WMI'01*, July 2001.

[9] A. Capone, M. Gerla and R. Kapoor, "Efficient Polling Schemes for Bluetooth Picocells", *Proc. IEEE ICC'01*, June 2001.

[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", *Proc. IEEE INFOCOM'01*, Apr. 2001.

[11] L. Fratta, M. Gerla and K. Kleinrock, "The Flow Deviation Method: an Approach to Store-and-Forward Communication Network Design", *Networks*, Vol. 3, pp. 97-133, 1973.

[12] M. Gerla, J.A.S. Monteiro and R. A. Pazos-Rangel, "Topology Design and Bandwidth Allocation in ATM Nets", *IEEE J. on Selected Areas in Comm.*, Vol. 7, pp. 1253-1262, Oct. 1989.

[13] N. Johansson, U. Korner and P. Johansson, "Performance Evaluation of Scheduling Algorithms for Bluetooth", *Proc. IFIP TC6 International Conf. on Broadband Communications*, Nov. 1999.

[14] P. Johansson, M. Kazantzidis, R. Kapoor and M. Gerla, "Bluetooth: An Enabler for Personal Area Networking", *IEEE Network*, pp. 28-36, Sep/Oct. 2001.

[15] L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay*, McGraw-Hill, New York, 1964.

[16] C. Law, A. M. Mehta and K.Y. Siu, "Performance of a New Bluetooth Scatternet Formation Protocol", *Proc. ACM MOBIHOC'01*, Oct. 2001.

[17] L. Lovasz and M. D. Plummer, "Matching Theory", *Annals of Discrete Mathematics*, 29, North Holland, 1986.

[18] B. A. Miller and C. Bisdikian, *Bluetooth Revealed*, Prentice Hall, 2000.

[19] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley and Sons, 1988.

[20] R. A. Pazos-Rangel and M. Gerla, "Express Pipe Networks", *Proc. Global Telecommunications Conf*., pp. B2.3.1-5, 1982.

[21] A. Racz, G. Miklos, F. Kubinszky and A. Valko, "A Pseudo Random Coordinated Scheduling Algorithm for Bluetooth Scatternets", *Proc. ACM MOBIHOC'01*, Oct. 2001.

[22] I. A. Saroit Ismail, "Bandwidth Problems in High-Speed Networks", *IBM J. of Resarch and Development*, Vol. 44, No. 6. Nov. 2000.

[23] A. Segall, "Optimal Virtual Routing for Virtual Line-Switched Data Networks", *IEEE Trans. on Comm.*, Vol. 27, pp. 201-209, Jan. 1979.

[24] G. V. Zaruba, S. Basagni and I. Chlamtac, "Bluetrees – Scatternet Formation to Enable Bluetooth-Based Ad Hoc Networks", *Proc. IEEE VTC'01-Spring*, May 2001.