Analysis of Bandwidth Allocation Algorithms for Wireless Personal Area Networks

Randeep Bhatia * Adrian Segall † Gil Zussman ‡

Abstract

A major issue in the design and operation of ad hoc networks is sharing the common spectrum among links in the same geographic area. Bandwidth allocation, to optimize the performance of networks in which each station can converse with at most a single neighbor at a time, has been recently studied in the context of Bluetooth Personal Area Networks. There, centralized and distributed, capacity assignment heuristics were developed, with applicability to a variety of ad hoc networks. Yet, no guarantees on the performance of these heuristics have been provided. In this paper, we extend these heuristics such that they can operate with general convex objective functions. Then, we present our analytic results regarding these heuristics. Specifically, we show that they are β-approximation ($\beta < 2$) algorithms. Moreover, we show that even though the distributed and centralized algorithms allocate capacity in a different manner, both algorithms converge to the same results. Finally, we present numerical results that demonstrate the performance of the algorithms.

1 Introduction

In the last four decades, much attention has been given to the research and development of bandwidth allocation and scheduling schemes for wired and wireless networks [6],[23]. The bandwidth allocation problem in wireless ad hoc networks significantly differs from the problem in static communication networks. For instance, one of the major problems in the design and operation of ad hoc networks is sharing the common spectrum among links in the same geographic area. A unified framework for dealing with many variations of this problem has been presented in [23]. In this paper, we focus on bandwidth allocation in networks in which *each station can converse with at most a single neighbor at a time*. Namely, we focus on networks in which the set of active links at any point of time constitutes a matching[1] in the network graph.

This bandwidth allocation problem has been studied by Hajek and Sasaki [11], who proposed centralized algorithms for finding a minimum length schedule to satisfy given traffic requirements. The problem has been recently revisited mainly due to the emergence of Bluetooth (IEEE 802.15.1) Personal Area Networks (PANs) [8] and its solution is relevant to other technologies in which a node can communicate with at most a single

*Bell Labs, Lucent Technologies, Murray Hill, NJ 07974. E-mail: *randeep@research.bell-labs.com.*

†Department of Electrical Engineering, Technion, Haifa 32000, Israel. E-mail: *segall@ee.technion.ac.il.*

‡Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139. E-mail: *gilz@mit.edu.*

[1]A set of links is a matching, if no two links are incident to the same node.

To appear in ACM/Springer Wireless Networks (WINET), Special issue of selected papers from WiOpt'04, 2005

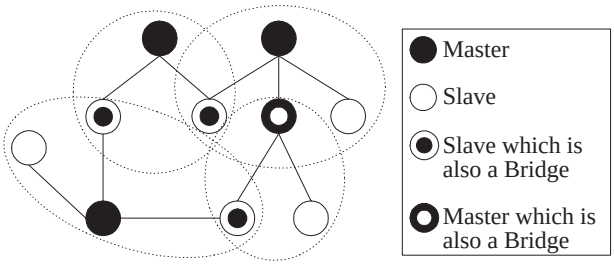

Figure 1: An example of a Bluetooth scatternet composed of 4 piconets.

neighbor at a time (such as IEEE 802.15.3 [15]).

In this paper we mainly focus on bipartite network graphs, rather than on networks operating according to a specific technology. Yet, since most of the recent research regarding the considered problem has focused on PANs, we briefly review the the main characteristics of Bluetooth and IEEE 802.15.3. Bluetooth enables portable mobile devices to connect and communicate wirelessly via short-range ad-hoc networks. 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 uses a slotted scheme where the only allowed communication is between a master and a slave and the master-to-slave and slave-to-master transmissions happen in alternate slots. Connected piconets in the same geographic area form a *scatternet* (see for example Fig. 1). In a scatternet, 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 refer to such a unit as a *bridge*).

Another PAN technology is IEEE 802.15.3 [15], whose Medium Access Control (MAC) standard is expected to be used as one of the Ultra WideBand MAC standards [19],[22]. The basic topology (piconet) is a collection of *devices* (DEVs) operating together with one *PicoNet Coordinator* (PNC), which allocates network resources. The timing is based on a superframe which is comprised of three main sections: the beacon, the optional Contention Access Period, and the Channel Time Allocation Period (Fig. 2 illustrates the structure of the superframe). The beacon is used to carry control information to the entire piconet. The Contention Access Period is optional and is managed according to a CSMA/CA mechanism. The Channel Time Allo-

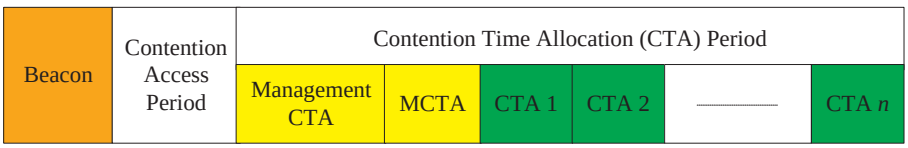

Figure 2: The structure of a superframe in IEEE 802.15.3.

cation Period is composed of Channel Time Allocations (CTAs), that are used for isochronous streams and asynchronous data connections, as well as Management CTAs (MCTAs) that are used for commands. The CTAs are allocated by the PNC according to the DEVs requests and can be used for PNC-DEV communication as well as for inter-DEV communication. The IEEE 802.15.3 standard supports the operation of a few piconets in the same geographic area. However, the mechanisms for inter-piconet communication are out of the scope of the standard.

Efficient network operation requires determining the capacities that should be allocated to each link (e.g. portion of slots in Bluetooth or allocation of CTAs in IEEE 802.15.3), such that the network performance is optimized. The need to find a feasible capacity allocation in Bluetooth scatternets has been identified by Baatz et al. [3]. In [29] the scatternet capacity assignment problem has been formulated as a problem of minimizing a convex function over a convex set contained in the matching polytope (similar formulations appear in [2] and [26]) and optimal as well as heuristic algorithms for its solution have been proposed. Sarkar and Tassiulas [25],[26] have studied the problem of maxmin fair allocation of bandwidth in networks in which a node can converse with at most a single neighbor at a time (similarly to the case in Bluetooth and in specific IEEE 802.15.3 networks). Finally, [9] extended the model of Hajek and Sasaki [11] by replacing the network graph with a SINR condition (i.e. links can be simultaneously active, if a given signal to interference and noise ratio is exceeded when transmitters use optimally chosen transmit powers).

The work in [29] presents distributed and centralized heuristics for the solution of the capacity assignment problem in Bluetooth scatternets, but does not include a detailed analysis of the performance of the algorithms. In addition, these heuristic algorithms have been developed under the assumption that Kleinrock's approximation assumption holds. As argued by Sarkar and Tassiulas [25],[26], although the algorithms have been developed in the context of Bluetooth scatternets, they can be applied to any ad hoc network in which a node transmits to a single neighbor at a time, and in which multiple transmissions can take place as long as they do not share a common node. Therefore, as mentioned above, in this paper we focus on network graphs and not necessarily on Bluetooth scatternets. We make similar assumption to the ones made in [29]. Namely, the analysis is based on a static model with stationary flows and unchanging topology. We also assume that the network graph is bipartite[2] and that the capacities allocated to the links are the total bi-directional capacities. This assumption holds in many networks of the considered type. Specifically, in [7] and [29] it is shown that Bluetooth scatternet topologies which are nonbipartite may result in poor bandwidth utilization.

First, we extend the heuristics, presented in [29], such that they can operate with general convex objective

[2] 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.

functions. We then show that an optimal solution (to a degree of accuracy of $1/M$, where M is a large constant) can be obtained in polynomial time. Yet, since in wireless networks there is no central authority responsible for network optimization and since the polynomial algorithms do not easily lend themselves to distributed implementation, we focus on the heuristic algorithms.

We analyze the performance of these heuristics and show that they are actually β-approximation ($\beta < 2$) algorithms for the solution of the capacity assignment problem. An important property regarding the tightness of the upper bound on their performance is also presented. Then, we show that although they allocate capacity in a different order, the distributed algorithm converges to the same results as the centralized one. Finally, numerical results are presented.

As mentioned above, Hajek and Sasaki [11] have proposed centralized algorithms for finding a minimum length schedule to satisfy given traffic requirements. In our context, the scheduling problem studied in [11] can be viewed as a feasibility test. Namely, given a set of capacities (larger than the required flows), the algorithm provided in [11] can verify whether or not these capacities could be allocated (without violating the capacity constraints of the nodes and the network). Recently, Sarkar and Tassiulas [25],[26] have proposed maxmin fair scheduling algorithms for a similar setting. We extend these models and try to obtain a set of capacities which is not only feasible but also *minimizes a general non-linear objective function* (e.g. average delay). The main contribution of this paper is a rigorous analysis of this problem as well as of centralized and distributed approximation algorithms for its solution.

In the specific context of Bluetooth scatternets, we view bandwidth allocation algorithms as the missing link between topology construction and scheduling algorithms (see the discussion in [29]). Numerous topology construction and scheduling algorithms for Bluetooth scatternets have been recently proposed (see for example the reviews in [4] and [27]). Since analysis of algorithms tailored for Bluetooth scatternets has been done mostly via simulation, an analytical approach that provides rigorous bounds on the performance is of great importance. Therefore, throughout the paper, we try to map the connections between the analyzed algorithms and Bluetooth scatternets. In addition, we note that the implementation of the algorithms in networks based on IEEE 802.15.3 will be possible only in the case inter-DEV communication is not used and DEVs do not operate as PNCs.[3]

This paper is organized as follows. Section 2 presents the model and the formulation of the problem. In Section 3, we show that the problem can be solved by centralized polynomial-time algorithms. The heuristic algorithms presented in [29] are reviewed and extended for a general objective function in Section 4. In

[3]If inter-DEV communication is allowed or if a DEV can also be a PNC, the set of active links does not necessarily constitutes a matching in a bipartite graph.

Section 5, we show that the heuristic algorithms are β-approximation ($\beta < 2$) algorithms. In Section 6, we show that the distributed and centralized algorithms converge to the same results. Section 7 presents numerical results and Section 8 summarizes the results.

2 Model and Problem Formulation

2.1 Model and Preliminaries

We model the network by an undirected bipartite graph $G = (N, L)$, where N denotes the collection of *nodes* $\{1, 2, \ldots, n\}$ and L denotes the set of *bi-directional links*. We denote by $I(i)$ the set of links $e = (i, j) \in L$ that are incident on node i. We concentrate on the total bi-directional link capacity. Hence, we assume that the performance of a link (e.g. the average packet delay) is a function of the total link flow and of the total link capacity. Let F_e be the average bi-directional flow on link e and let C_e be the capacity of link e (the units of F and C are bits/second). We assume that the average bi-directional flow is positive on every link ($F_e > 0 \ \forall e \in L$). We define f_e as the ratio between F_e and the maximal possible flow on a link [4]. We also define c_e as the ratio between C_e and the maximal possible capacity of a link. It is obvious that $0 < f_e \leq 1$ and that $0 < c_e \leq 1$. We shall refer to f_e as the *flow on link* e and to c_e as the *capacity of link* e.

The objective of the capacity assignment algorithms, analyzed in this paper, is to minimize the sum of non-linear decreasing functions of the links' capacities. The function of the capacity of link e shall be denoted by D_e. We assume that the flow rates (f_e) are given and that D_e is a function of the link capacity c_e only. The derivative of $D_e(c_e)$ is denoted by $D'_e(c_e)$. We assume the following properties of the function $D_e(\cdot)$

Definition 1 *$D_e(\cdot)$ is defined such that all the following hold:*

- *D_e is a nonnegative continuous decreasing function of c_e with continuous first and second derivatives.*
- *D_e is convex.*
- $\lim_{c_e \to f_e} = \infty$.

One possible application of the analyzed algorithms is to minimize the average delay in the network. Another application is to attain proportional fair [17] capacities. In case one wishes to minimize the average delay, D_e should be defined as the total delay per unit time of all traffic passing through link e. Since the total traffic in the network is independent of the capacity assignment procedure, we can minimize the average delay by minimizing the total delay (i.e. by minimizing $\sum_{e \in L} D_e(c_e)$). *For convenience*, we shall refer to the

[4] For example, currently the maximal flow on a symmetrical *Bluetooth* link is 867.8 Kb/s.

function that we wish to minimize as the *delay*. We note that the properties presented in Definition 1 conform to the analytic and simulation results regarding packet delay in Bluetooth piconets presented in [12],[21],[30], and [31].

Alternatively, if one wishes to provide some degree of proportional fairness to the links, the following definition of D_e can be used[5]:

$$D_e(c_e) = \begin{cases} -w_e \log(c_e - f_e) & c_e > f_e \\ \infty & c_e \leq f_e \end{cases}. \quad (1)$$

where w_e is a positive constant.

The heuristic algorithms in [29] have been developed assuming that the delay function is based on *Kleinrock's independence approximation* [18] which is described in the following definition:

$$D_e(c_e) = \begin{cases} f_e/(c_e - f_e) & c_e > f_e \\ \infty & c_e \leq f_e \end{cases}. \quad (2)$$

We will show that these algorithms can be extended for the general functions described in Definition 1. This extension is important, since it allows using the algorithms with different *delay* functions as well as with proportional fairness functions. We note that in order to obtain *numerical results* and to provide some insight regarding the performance of the algorithms, we shall use Kleinrock's approximation[6].

2.2 Formulation of the Problem

The objective of the capacity assignment algorithms is to minimize the sum of non-linear decreasing functions of the capacity (i.e. the functions - $D_e(c_e)$). Accordingly, the problem of *capacity assignment in bipartite graphs* (CAB) is formulated as follows [29][7].

Problem CAB

Given: Topology of a bipartite graph and flows (f_e).

[5]Within the feasibility region that shall be defined in Section 2.2, (1) conforms to Definition 1

[6]Kleinrock's independence approximation has been shown to provide a relatively good estimation for the delay in networks involving Poisson stream arrivals. Therefore, it is used for the *numerical* evaluation of approximation capacity assignment algorithms.

[7]In [29] the problem is referred to as *scatternet capacity assignment in bipartite graphs* (SCAB). In order to emphasize the generalization of the analysis, we shall omit the term *scatternet* from the names of the problem and the algorithms.

Objective: Find capacities (c_e) such that the average delay is minimized:

$$\text{minimize} \quad \sum_{e \in L} D_e(c_e) \tag{3}$$

$$\text{subject to:} \quad c_e > f_e \ \forall e \in L \tag{4}$$

$$\sum_{e \in I(i)} c_e \leq 1 \ \forall i \in N \, . \tag{5}$$

The second set of constraints (5) reflects the fact that the total capacity of the links connected to a node cannot exceed the maximal link capacity. Constraints similar to (5) appear in problems formulated in [2],[25], and [26]. Henceforth we will assume that there is a feasible solution to Problem CAB. We note that the formulation of the problem is based on the assumption that the flow rates are given by higher layer protocols, based on the traffic statistics.

Notice that based on the Edmonds' Theorem [10] and the analysis of Hajek and Sasaki [11], for *nonbipartite graphs* it is sufficient to replace (5) by

$$\sum_{e \in I(i)} c_e \leq 2/3 \ \forall i \in N \, . \tag{6}$$

Although (6) is a sufficient condition for feasible capacity allocation in nonbipartite graphs, it is not a necessary condition, and therefore the obtained capacity allocation may not be optimal.

We note that when considering Bluetooth scatternets, one might want to take into account the time that is wasted in the process of moving a bridge from one piconet to another (known as the guard time). In (5) we neglect these guard times. In case they should be taken into account, (5) should be replaced with:

$$\sum_{e \in I(i)} c_e \leq 1 \quad \forall i \in N - B$$

$$\sum_{e \in I(i)} c_e \leq b \quad \forall i \in B \, ,$$

where B is the set of bridges and b is the normalized capacity available to links connected to a bridge ($b < 1$).

In order to present an equivalent formulation of Problem CAB, we define the *slack capacity* of a node as follows:

Definition 2 *The* slack capacity of node i *is the maximum capacity which can be added to links connected to the node. It is denoted by s_i and is given by $s_i = 1 - \sum_{e \in I(i)} c_e$.*[8]

[8] In case the algorithms are applied to nonbipartite graphs (i.e. (5) is replaced by (6)), s_i should be defined as $2/3 - \sum_{e \in I(i)} c_e$. Similarly, in case one wishes to take into account the guard times in Bluetooth scatternets, s_i should be defined as $b - \sum_{e \in I(i)} c_e$ for bridge nodes.

In both algorithms, considered in this paper, all link capacities are initially equal to the flows on the links ($c_e = f_e,\ \forall\, e \in L$). Therefore, by definition, initially (before the first phase of any algorithm) $s_i = 1 - \sum_{e \in I(i)} f_e$. Using the definition of s_i and setting $\tau_e = c_e - f_e, e \in L$, we obtain the following formulation of Problem CAB:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in L} D_e(\tau_e) \\ \text{subject to:} \quad & \sum_{e \in I(i)} \tau_e \leq s_i \; \forall i \in N \\ & \tau_e > 0 \; \forall e \in L \; . \end{aligned} \tag{7}$$

We denote by τ_e^* the optimal solution to (7). As mentioned before, we assume that the problem has a feasible solution and therefore, s_i must be greater than zero for all $i \in N$. Thus, $0 < s_i \leq 1$ for all i. Additionally, since by definition $f_e > 0$ and $c_e \leq 1$, $\tau_e < 1$, for all e.

3 Transformation to a Flow Problem

In this section we show that Problem CAB can be transformed to a minimum cost flow problem with a separable convex objective function. The transformation is based on creating a super-source and a super-sink as described in Fig. 3 and on using directional edges instead of bi-directional edges. The super-source is connected by directed edges to all the nodes in one of the sets composing the bipartite graph. The capacities of these edges are defined as the slack capacities (s_i) of the nodes to which they are connected. Similarly, the nodes in the other set are connected to a super-sink by edges with limited capacity. The edges connecting the nodes of the bipartite graph have unlimited capacity. The super-sink and super-source are also connected by an edge with unlimited capacity. The flow values in the transformed network are τ_e. The cost of a flow τ_e traversing an edge e connecting the nodes of the bipartite graph is $D_e(\tau_e)$. The cost of a flow on any other edge is 0. A minimum cost flow in the transformed network provides a solution to Problem CAB.

The flow problem presented above is a specific case of the *convex cost flow problem* discussed by Ahuja et al. [1, p. 556]. In [1] a polynomial-time algorithm for finding an *integer* solution to the minimum cost flow with convex separable objective function is provided. If one needs to obtain a more accurate solution than the integer solution (as required in our case), he could substitute each τ_e by $y_e = M\tau_e$, for sufficiently large value of M, and adjust the objective function and constraints accordingly. If y_e^* denotes an integer optimal solution of the transformed problem, $\tau_e^* = y_e^*/M$ is an optimal solution of the original problem (to a degree of accuracy of $1/M$).

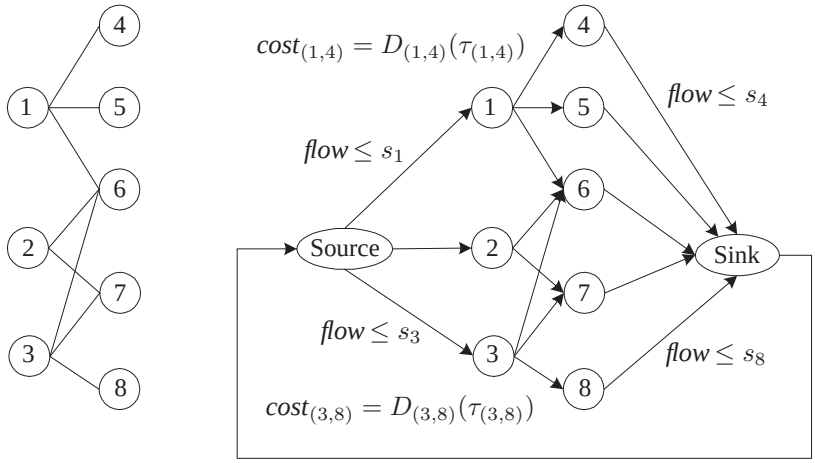

Figure 3: An example of the transformation of a bipartite graph to a flow network.

The method presented in [1] is a variant of the algorithm presented by Minoux [20] and it is in the framework of the scaling algorithm given by Hochbaum and Shanthikumar [14]. Another algorithm for the minimum cost flow problem, which is based on the framework of [14], was presented in [13]. Finally, Karzanov and McCormick [16] studied a problem which generalizes the minimum non-linear cost circulation problem and provided algorithms that differ from the algorithms provided in [14]. Their polynomial-time algorithm for obtaining an integer solution can be used in order to obtain a solution to Problem CAB.

To conclude, there are a couple of polynomial algorithms that can be used for obtaining an integer solution to Problem CAB, thereby providing a non-integer solution (to a degree of accuracy of $1/M$). However, these algorithms cannot be easily modified in order to allow distributed implementation as required in wireless networks. *Therefore, in this paper we focus on approximation algorithms that can be easily implemented in a distributed manner*.

4 Approximation Algorithms

A *capacity assignment algorithm* has to obtain a solution to Problem CAB (e.g. to determine what portion of the slots should be allocated to each link). In this section we briefly review the *centralized* and *distributed* approximation algorithms for bipartite graphs, presented in [29]. We shall refer to these algorithms as the *heuristic centralized/distributed capacity assignment* algorithms (Algorithm HCCA/HDCA respectively). Since these algorithms have been developed with an objective function based on Kleinrock's approximation (2), *we extend them here for a general objective function*, based on the delay function described in Definition 1.

In both algorithms, all link capacities are initially equal to the flows on the links ($c_e = f_e,\ \forall\, e \in L$). The

algorithms select a node, allocate the slack capacity to the links connected to it, whose capacities are equal to the flows, and update the slack capacities of the neighboring nodes. Then, another node is selected, capacity is allocated, the slack capacities are updated, and so on. In both algorithms the nodes are selected according to their *delay derivatives*, which are computed in the following manner.

Every node i, that has to compute its delay derivative, obtains the optimal solution to the following local optimization problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in I(i), c_e = f_e} D_e(\tau_e) \\ \text{subject to:} \quad & \sum_{e \in I(i), c_e = f_e} \tau_e \leq s_i \\ & \tau_e > 0 \; \forall e \in I(i),\, c_e = f_e \,. \end{aligned} \tag{8}$$

This solution determines how the slack capacity of this node should be allocated to those adjacent links, whose capacities have not yet been assigned, in case the node would be selected for allocating capacity. The problem described in (8) is an optimization of a convex function over a simplex [5, p. 359] (notice that in the optimal solution, the first constraint must hold with equality). Therefore, it has a unique optimal solution that can be obtained by using the method of Lagrange Multipliers and solving the following system of equations:

$$D'_e(\tau^*_e) = \lambda^*_i \quad \forall e \in I(i), c_e = f_e \tag{9}$$

$$\sum_{e \in I(i), c_e = f_e} \tau^*_e = s_i \tag{10}$$

Since the number of node's i neighbors ($|I(i)|$) is usually bounded by a small constant (e.g. 7 in Bluetooth), and since there is only one constraint in the non-linear program (8), the solution to any degree of accuracy ϵ (for a fixed constant ϵ) can be found in constant time. In the specific case that Kleinrock's independence approximation is assumed, the optimal solution follows the square root assignment [18, p. 20]:

$$\tau^*_e = \frac{s_i \sqrt{f_e}}{\sum\limits_{l \in I(i),\, c_l = f_l} \sqrt{f_l}} \;\; \forall e \in I(i), c_e = f_e \,. \tag{11}$$

After solving (9)-(10) a node computes its delay derivative according to the following definition.

Definition 3 *The* delay derivative of node i *is denoted by* d_i *and is given by:* $d_i = |\lambda^*_i|$.

For example, under Kleinrock's approximation[9]:

$$d_i = \left(\frac{\sum_{e \in I(i),\, c_e = f_e} \sqrt{f_e}}{s_i} \right)^2 . \tag{12}$$

After a node k is selected, *the slack capacity of this node is allocated to those adjacent links, whose capacities have not yet been assigned*, according to the solution of (8). Capacity is allocated to a link only once. Hence, we define a *fully allocated node* as a node whose all adjacent link capacities have been assigned[10]. Accordingly, a *non-fully allocated node* is a node which has at least one adjacent link whose capacity has not been assigned.

4.1 Centralized Algorithm (Algorithm HCCA)

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. Algorithm HCCA, which is based on this methodology, is described in Fig. 4. The input is the topology and the flows (f_e), and the output is the link capacities: c_e. It can be seen that the complexity of the algorithm is $O(n^2)$, since as mentioned earlier for all i the d_i values can be computed in constant time to any fixed degree of accuracy. Notice that $O(n^2)$ is about the complexity of a single iteration in the optimal algorithm, presented in [29].

1 **set** $c_e = f_e \ \forall e \in L$
2 **compute** $d_i \ \forall i \in N \bigcap i$ non-fully allocated
3 **set** $k = \underset{i \in N \bigcap i \text{ non-fully allocated}}{\arg\max} d_i$
4 **set** $c_e =$ optimal solution of (8) $\forall\, e : e \in I(k),\, c_e = f_e$
5 **if** there exists $e \in L$ such that $c_e = f_e$
6 **then** go to 2
7 **else** stop

Figure 4: Algorithm HCCA for obtaining an approximate solution to Problem CAB.

[9] We note that in [29], where kleinrock's approximation was always assumed, d_i was defined as the square root of its current value under this assumption (i.e $d_i = \sum_{e \in I(i),\, c_e = f_e} \sqrt{f_e}/s_i$). This modification does not affect the performance of the algorithms. Yet, it simplifies their analysis.

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

4.2 Distributed Algorithm (Algorithm HDCA)

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. Once a node receives the token, it can either allocate its slack capacity or decide to send the token to a neighbor. The assignment of slack capacity is the same as in the centralized algorithm. However, the selection of the node which holds the token and the decision 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.[11] The node decides to either 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 that popped its details from the stack. 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 algorithm halts when all the neighbors are fully allocated and the stack is empty (it always terminates at the initiating node).

Fig. 5 presents the pseudocode of the procedure executed by a node in the allocation state. Unlike the centralized algorithm in which a node allocates capacity, if its d_i is the largest in the network, in the distributed algorithm a node allocates capacity, if it holds the token and its d_i is larger than the d_is of its neighbors.

Fig. 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.

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

```
1  push into the parents stack the details of the node which sent the token
2  find the node with the largest d_i 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 the optimal solution of (8)
7        update the neighbors
8        change the state to token transfer state
```

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

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

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

5 Approximation Ratios

In this section we show that Algorithm HCCA is β-approximation ($\beta < 2$) algorithm for the solution of Problem CAB. First, we present a new algorithm for the solution of Problem CAB. Then, we prove that the new algorithm outperforms any 2-approximation algorithm. Finally, we prove that Algorithm HCCA obtains results which are equal or better than the results obtained by the new algorithm. We note that in this section we also present an interesting property, of the upper bound, on the performance of the algorithms.

Recall that in (8) we have defined a set of $|N|$ non-linear programs, one for each $i \in N$. We denote the optimal solution of (8) for node i by τ_e^i and define the corresponding optimal value OPT_i of the objective function as

$$OPT_i = \sum_{e \in I(i)} D_e(\tau_e^i) \, . \tag{13}$$

We now present a simple algorithm, referred to as *Algorithm ACA* (Approximate Capacity Assignment), for obtaining an approximate solution to the Problem CAB. We denote by $\hat{\tau}_e$ the solution obtained by Algorithm ACA. For every node i, the algorithm computes the optimal solution to (8) (i.e. $\tau_e^i, \forall i \in N, e \in I(i)$). Then, it sets for all $e = (i, j) \in L$:

$$\hat{\tau}_e = \min(\tau_e^i, \tau_e^j). \tag{14}$$

We first show that $\hat{\tau}_e$ is a feasible solution to Problem CAB (i.e. to (7)). It is easy to see that $\hat{\tau}_e > 0, \forall e \in L$ and by construction $\hat{\tau}_e \leq \tau_e^i$. Since τ_e^i is an optimal solution to (8), for node i we have $\sum_{e \in I(i)} \tau_e^i \leq s_i$ and hence $\sum_{e \in I(i)} \hat{\tau}_e \leq s_i$.

We now show that Algorithm ACA is better than a 2-approximation algorithm for the problem CAB. In order to prove it, we first present the following lemma[12].

Lemma 1 (Zussman and Segall, 2004 [29]) *If $d_j > d_i$, then $\tau_{ij}^j < \tau_{ij}^i$. If $d_j = d_i$, then $\tau_{ij}^j = \tau_{ij}^i$.*

Theorem 1 *Algorithm ACA is a β-approximation algorithm ($\beta < 2$) for Problem CAB.*

Proof: For any node $i \in N$, τ_e^* (the optimal solution to the non-linear program (7)) is a feasible solution to the non-linear program (8). Hence, since OPT_i is the optimal value of the objective function for (8):

$$\sum_{e \in I(i)} D_e(\tau_e^*) \geq OPT_i \; \forall i \in N. \tag{15}$$

Adding up for all $i \in N$ and noting that the term for each edge appears exactly twice in the sum we have

$$2 \sum_{e \in L} D_e(\tau_e^*) \geq \sum_{i \in N} OPT_i. \tag{16}$$

For each node i we define a set of incident edges $J(i) \subseteq I(i)$ as those edges $e = (i, j) \in L$ for which $\tau_e^i < \tau_e^j$, or in the case of a tie ($\tau_e^i = \tau_e^j$), the node index $i < j$. It is easy to see that $J(i)$ forms a partition of the set of edges L. Thus:

$$\sum_{e \in L} D_e(\hat{\tau}_e) = \sum_{i \in N} \sum_{e \in J(i)} D_e(\hat{\tau}_e) = \sum_{i \in N} \sum_{e \in J(i)} D_e(\tau_e^i) \leq \sum_{i \in N} OPT_i,$$

where the last inequality follows from the definition of OPT_i in (13) and the fact that $J(i) \subseteq I(i)$. However, for a more tighter analysis we make the following observation. From Lemma 1 it follows that the node j with the smallest delay derivative d_j (in case of a tie, j is selected to be the node with the largest index) must have that $J(j) = \emptyset$. Thus, there is at least one node j with $J(j) = \emptyset$, and therefore since by definition $OPT_j > 0$ we have

$$\sum_{e \in L} D_e(\hat{\tau}_e) \leq \sum_{i \in N, i \neq j} OPT_i < \sum_{i \in N} OPT_i$$

Combining this with (16) we get

$$\sum_{e \in L} D_e(\hat{\tau}_e) < 2 \sum_{e \in L} D_e(\tau_e^*),$$

[12]Notice that [29] shows that the lemma holds for delay functions following Kleinrock's approximation assumption. It is easy to see that it can be extended to delay functions with the properties defined in Definition 1.

thus showing that Algorithm ACA is a β-approximation ($\beta < 2$) algorithm for the Problem CAB. □

Algorithm HCCA (described in Section 4.1) can be described as follows. Let $\bar{\tau}_e$ be the solution obtained by Algorithm HCCA. At any phase, the algorithm starts out with a graph (initially set to the original graph G) with slack capacities s_i (initially set to the original graphs slack capacities). In this graph, it finds the node i with the maximal value of d_i, solves the non-linear program (8) for that node i optimally, and sets

$$\bar{\tau}_e = \tau_e^i \ \ \forall e \in I(i).$$

It then decreases the slack capacities s_j for every node j, which is a neighbor of i, by the sum of $\bar{\tau}_e, e \in I(j)$ for all $\bar{\tau}_e$ that get set in this phase. This is done to reflect the capacity that has been already allocated. Any node i with $s_i = 0$ is removed from the graph. Also all the edges e that are assigned a value τ_e in this phase are removed from the graph. The new graph and the new slack capacities become input for the next phase. The algorithm terminates when no more edges are left in the graph.

Let L^p be the set of edges such that $\bar{\tau}_e,\ e \in L^p$ is set in phase p. Let ${\tau_e^i}^p$ be the optimal solution obtained for the non-linear program (8) of node i for the graph G^p used by Algorithm HCCA in phase p along with the slack capacities s_i^p. Let the node delay derivatives in phase p be d_i^p. Note that $G^1 = G$, $s_i^1 = s_i, \forall i \in N$, $d_i^1 = d_i, \forall i \in N$, and ${\tau_e^i}^1 = \tau_e^i, \forall e \in I(i)$. Let $I^p(i)$ be the set of edges incident on node i in G^p. Due to Lemma 1 and the fact that at each phase p, the node with the largest d_i^p is selected, at the end of phase p we have

$$\bar{\tau}_e = \min\left({\tau_e^i}^p, {\tau_e^j}^p\right) \ \ \forall e = (i,j) \in L^p. \tag{17}$$

We now show that for each edge the delay obtained by the algorithm HCCA is at most the delay obtained by the Algorithm ACA. In order to prove it, we first prove the following lemma.

Lemma 2 *${\tau_e^i}^{p+1} \geq {\tau_e^i}^p$ for all nodes i and edges $e \in I^{p+1}(i)$.*

Proof: Consider a node i that does not have edges whose capacities are set at phase p. For that node, ${\tau_e^i}^{p+1} = {\tau_e^i}^p$. Now consider a node i that does not have the maximal d_i at phase p but which has at least one edge whose capacity is set at phase p (i.e. some of the edges in $I^p(i)$ are in L^p). In the optimal solution of (8) at phase p

$$\sum_{e \in I^p(i)} {\tau_e^i}^p = s_i^p \,. \tag{18}$$

Similarly in the optimal solution at phase $p+1$

$$\sum_{e \in I^{p+1}(i)} {\tau_e^i}^{p+1} = s_i^{p+1} \,. \tag{19}$$

Since

$$s_i^{p+1} = s_i^p - \sum_{e \in I^p(i) \bigcap L^p} \bar{\tau}_e$$

and since according to (17) for $e = (i,j) \in L^p$, $\bar{\tau}_e \leq \tau_e^{i\,p}$, it follows from (18) and (19) that:

$$\sum_{e \in I^{p+1}(i)} \tau_e^{i\,p+1} \geq \sum_{e \in I^{p+1}(i)} \tau_e^{i\,p} \ . \tag{20}$$

According to Definition 1 and since $c_e = \tau_e + f_e$, $D'_e(\tau_e)$ is an increasing function. Due to (20) at least for one link connected to i, $\tau_e^{i\,p+1} \geq \tau_e^{i\,p}$. Therefore, for that link $D'_e(\tau_e^{i\,p+1}) \geq D'_e(\tau_e^{i\,p})$. According to (9) at each phase p, the values of $D'_e(\tau_e^{i\,p})$ are equal for all $e \in I^p(i)$. Thus,

$$D'_e(\tau_e^{i\,p+1}) \geq D'_e(\tau_e^{i\,p}) \ \forall e \in I^{p+1}(i)\,.$$

Consequently, $\tau_e^{i\,p+1} \geq \tau_e^{i\,p}\ \forall e \in I^{p+1}(i)$. □

Proposition 1 *Algorithm HCCA is a β-approximation algorithm ($\beta < 2$) for the Problem CAB.*

Proof: We show that $\bar{\tau}_e \geq \hat{\tau}_e, \forall e \in L$ [13], thus showing that the value of the objective function of the non-linear program (7) for the solution $\bar{\tau}_e$ is at most the value of the objective function of (7) for the solution obtained by a β-approximation algorithm ($\beta < 2$). It follows from Lemma 2 that for any node i and link $e \in I^p(i)$:

$$\tau_e^{i\,p} \geq \tau_e^{i\,p-1} \ldots \geq \tau_e^{i\,1} = \tau_e^i.$$

Thus, $\tau_e^{i\,p} \geq \tau_e^i$ and $\tau_e^{j\,p} \geq \tau_e^j$ for a link $e = (i,j) \in L^p$. Hence, since for such a link $e = (i,j)$, we have $\bar{\tau}_e = \min(\tau_e^{i\,p}, \tau_e^{j\,p})$ (see (17)) and $\hat{\tau}_e = \min(\tau_e^i, \tau_e^j)$ (see (14)), we have $\forall e = (i,j) \in L^p$:

$$\bar{\tau}_e = \min(\tau_e^{i\,p}, \tau_e^{j\,p}) \geq \min(\tau_e^i, \tau_e^j) = \hat{\tau}_e.$$ □

Finally, we show that any upper bound on the performance of Algorithm HCCA which is based on the relationships between OPT_i and the optimal solution (i.e. based on (16)) *is not tight*.

Proposition 2 *Assume that there exists $\beta\,(1 < \beta < 2)$ such that*

$$\sum_{e \in L} D_e(\bar{\tau}_e) \leq \frac{\beta}{2} \sum_{i \in N} OPT_i \leq \beta \sum_{e \in L} D_e(\tau_e^*), \tag{21}$$

thus implying that HCCA is a β-approximation algorithm, then there is no tight example in which the heuristic solution ($\bar{\tau}_e$), obtained by HCCA is exactly β times more than the optimal solution. In other words there is no

[13] Recall that $\bar{\tau}_e$ and $\hat{\tau}_e$ are the solutions obtained by algorithms HCCA and ACA, respectively.

example for which

$$\sum_{e \in L} D_e(\bar{\tau}_e) = \beta \sum_{e \in L} D_e(\tau_e^*). \tag{22}$$

Proof: Assume that an example in which (22) holds exists. This implies that all inequalities in (21) must hold with equalities. Specifically

$$\sum_{i \in N} OPT_i = 2 \sum_{e \in L} D_e(\tau_e^*).$$

Therefore, for any given node i, (15) holds with equality (i.e. $OPT_i = \sum_{e \in I(i)} D_e(\tau_e^*)$). Consequently, since the optimal solution for (8) is unique, we have for every link $e = (i, j)$: $\tau_e^i = \tau_e^* = \tau_e^j$. Thus, the solution obtained by Algorithm ACA ($\hat{\tau}_e$) is equal to the optimal solution (since $\hat{\tau}_e = \min\{\tau_e^i, \tau_e^j\}$). Finally, in the proof of Proposition 1, we have shown that $\bar{\tau}_e \geq \hat{\tau}_e, \forall e \in L$. Thus

$$\sum_{e \in L} D_e(\bar{\tau}_e) \leq \sum_{e \in L} D_e(\tau_e^*),$$

which contradicts (22). □

6 Convergence of the Distributed Algorithm

Due to the differences between the algorithms, the centralized algorithm (Algorithm HCCA) and the distributed algorithm (Algorithm HDCA) normally allocate capacity in different order. However, in this section we will show that *the two algorithms always converge to the same results*. Since in Section 5 we have shown that Algorithm HCCA is a β-approximation algorithm ($\beta < 2$), this implies that Algorithm HDCA has the same property. First, we show that the distributed algorithm halts after all the link capacities have been allocated. Then, we show that these link capacities are the same as the link capacities allocated by the centralized algorithm.

The proof that the distributed algorithm halts after all the link capacities have been allocated is based on the fact that the token either does not traverse a link or traverses it in both directions. Accordingly, 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. The formal proof is based on the following lemmas. The proofs of the lemmas appear in the Appendix.

Lemma 3 *In Algorithm HDCA, when a node i enters the token transfer state, it is fully allocated* ($c_e \neq f_e \; \forall e \in I(i)$).

Lemma 4 *When Algorithm HDCA halts, every node that has been in the allocation state has also been in the token transfer state.*

The proof of Lemma 4 implies that every node that has been in the allocation state has also executed Step 4 described in Fig. 6 (i.e. checked the status of the stack and then either halted or returned the token to a parent). Using the above lemmas we shall now prove the following proposition.

Proposition 3 *When Algorithm HDCA halts:* $c_e \neq f_e \; \forall e \in L$.

Proof: Assume that when the algorithm halts, there is a link e for which $c_e \neq f_e$ does not hold (either c_e is not defined or $c_e = f_e$). According to Lemma 3, nodes i and j do not enter the token transfer state during the execution of the algorithm. Consequently, according to Lemma 4, node i and j do not enter the allocation state during the execution. Since node j does not enter the allocation state, none of its neighbors ever executes Step 4 described in Fig. 6 (since before its execution they would send the token to node j which would enter the allocation state). Following the argument used in the proof of Lemma 4, due to the fact that the protocol halts, every node that has been in the allocation state has also executed Step 4 described in Fig. 6. Thus, none of the neighbors of j enters the allocation state and the capacities of the links connecting them to j are not assigned.

Using a similar argument, it can be shown that none of the link capacities of the neighbors of j are assigned. Consequently, no node enters the allocation state and no link capacity is assigned. This is a contradiction to the fact that the algorithm halts. □

We now need to show that the capacity allocated by Algorithm HDCA is equal to the capacity allocated by Algorithm HCCA. Thus, we first derive a property of the delay derivative of a node in algorithms HDCA and HCCA. Then, we prove by induction that the capacities allocated by the two algorithms are identical.

Lemma 5 *When a node i allocates capacity the delay derivatives (d_ks) of its non-fully allocated neighbors do not increase.*

Proof: Let k be a non-fully allocated neighbor of i. Let d_k^- and d_k^+ be its delay derivatives just before and just after (respectively) i allocates capacity. Denote by $E(k)$ the set of links that are incident on node k ($E(k) \subseteq I(k)$) such that $c_e = f_e$ just after i allocates capacity. Let $\tau_e^{k\,-}$ and $\tau_e^{k\,+}$ be the optimal solutions of node k to (8) just before and just after (respectively) i allocates capacity. According to (9) and Definition 3, for all $e \in E(k)$, $d_k^- = D'_e(\tau_e^{k\,-})$ and $d_k^+ = D'_e(\tau_e^{k\,+})$. According to Lemma 2, $\tau_e^{k\,+} \geq \tau_e^{k\,-} \; \forall e \in E(k)$. According to Definition 1, $D'_e(\tau_e)$ is an increasing function and therefore, $d_k^+ \geq d_k^-$. □

We note that if the Kleinrock's approximation (2) holds, the lemma results directly from the definition of

the delay derivative of a node (12) and the square root assignment (11). We present here a simple proof for that case. Let k be a non-fully allocated neighbor of i. Let d_k^- and d_k^+ be its delay derivatives just before and just after (respectively) i allocates capacity. Denote by $B(k)$ the set of links adjacent to k ($B(k) \subseteq I(k)$) such that $c_e = f_e$ just *before* i allocates capacity. According to (11) and (12), following the allocation:

$$d_k^+ = \left(\frac{\sum_{e \in B(k)} \sqrt{f_e} - \sqrt{f_{ik}}}{s_k - (c_{ik} - f_{ik})} \right)^2 = \left(\frac{\sum_{e \in B(k)} \sqrt{f_e} - \sqrt{f_{ik}}}{(\sum_{e \in B(k)} \sqrt{f_e})/\sqrt{d_k^-} - \sqrt{f_{ik}}/\sqrt{d_i^-}} \right)^2 . \quad (23)$$

Since i allocates the capacity, just before the allocation $d_i^- \geq d_k^-$. Accordingly,

$$d_k^+ \leq \left(\frac{\sum_{e \in B(k)} \sqrt{f_e} - \sqrt{f_{ik}}}{(\sum_{e \in B(k)} \sqrt{f_e})/\sqrt{d_k^-} - \sqrt{f_{ik}}/\sqrt{d_k^-}} \right)^2 = d_k^- . \quad (24)$$

Theorem 2 *The capacities (c_e) obtained by Algorithm HDCA are identical to the capacities obtained by Algorithm HCCA.*

Proof: According to Proposition 3, Algorithm HDCA halts only after all the link capacities have been allocated. Thus, we have to show that these link capacities are the same as the link capacities assigned by Algorithm HCCA.

We need to show that when Algorithm HDCA is executed, at any given time, the following properties hold:

1. For every non-fully allocated node j, the neighbors that allocate capacity after j in Algorithm HCCA are non-fully allocated (i.e. some or all neighbors which allocate capacity before it in Algorithm HCCA are fully allocated).

2. The values of the capacities that have already been allocated are the same as in Algorithm HCCA.

In order to prove it, we assume that the above properties hold at time t^- and we show that if a node i allocates capacity at time t (immediately after time t^-), these properties continue to hold.

At time t, Algorithm HDCA selects a non-fully allocated node i with a delay derivative (d_i) higher than the delay derivatives of its non-fully allocated neighbors and allocates capacity (Step 6 in Fig. 5). We wish to show that when i is selected, *all* (and not *some*) the neighbors which allocate capacity before it in Algorithm HCCA are fully allocated.

Denote by $d_i(t)$ the delay derivative of node i at time t. Denote the set of the non-fully allocated neighbors of i at time t^- by $M(i)$ ($m \in M(i)$, if $e \in I(i), e \in I(m)$, and $c_e = f_e$ at time t^-). let t_m be the time in which a node m allocates capacity in Algorithm HCCA. Due to the first property and due to Lemma 5, $d_i(t^-) \geq d_m(t^-) \geq d_m(t_m),\ \forall m \in M(i)$. In order for $m \in M(i)$ to allocate capacity before i in Algorithm HCCA, $d_m(t_m) \geq d_i(t_m)$ has to hold. However, in order for d_i to become smaller than d_m, one of the other non-fully allocated neighbors of i has to allocate capacity in Algorithm HCCA before time t_m. This cannot happen, since their delay derivatives are lower than the delay derivative of i. Thus, the nodes in $M(i)$ allocate capacity after i.

Since at time t, *all* the neighbors of i which allocate capacity before it in Algorithm HCCA are fully allocated, i allocates the same capacities as in Algorithm HCCA. Thus, at time t^+ (immediately after time t) the second property holds. Moreover, since we have shown that in Algorithm HCCA, i allocates capacity before the nodes in $M(i)$, at time t^+, the first property holds.

Finally, since the properties 1 and 2 hold before the first node allocates capacity, they also hold after the last node allocates capacity, and therefore, the capacity values allocated by Algorithm HDCA are identical to the values allocated by Algorithm HCCA. □

7 Numerical Results

In this section we present a few numerical examples that demonstrate the difference between the results obtained by the Algorithms HCCA and HDCA, the results obtained by Algorithm ACA, and the optimal results. In order to obtain numerical results, we assume in this section that the delay function follows Kleinrock's independence approximation (i.e. $D_e(\tau_e) = f_e/\tau_e$) and that the considered bipartite graphs are Bluetooth scatternets. First, we assume that the guard times are negligible, and therefore, no switching overheard is incurred by the bridges. Then, we present results for the case in which different nodes have different capacities (resulting, for example, from the switching overhead).

From the observations made in Section 5 it follows that:

$$\sum_{e\in L} D_e(\tau_e^*) \leq \sum_{e\in L} D_e(\bar{\tau}_e) \leq \sum_{e\in L} D_e(\hat{\tau}_e) < \sum_{i\in N} OPT_i \leq 2\sum_{e\in L} D_e(\tau_e^*)\,. \tag{25}$$

Namely: Optimal Solution $\leq$ Solution by HCCA $\leq$ Solution by ACA $< \sum_{i\in N} OPT_i \leq 2*$(Optimal Solution).

Fig. 7 illustrates a scatternet with given flow rates (the scatternet topology is based on the topology presented in [24, Fig. 4]). Table 1 presents the corresponding values of the measures presented in (25).

It can be seen that there is a small difference between the optimal and the approximate solution, obtained by Algorithm HCCA. A larger difference exists between the solutions obtained by Algorithm HCCA and Algorithm ACA. Furthermore, there is quite a large difference between the solution obtained by Algorithm ACA and its upper bound.

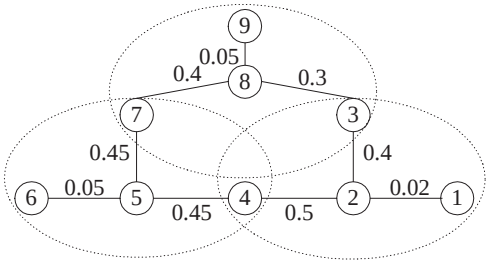

Figure 7: A scatternet with given flow rates.

Table 1: Optimal solution, approximate solutions, and the upper bound for the scatternet described in Fig. 7.

	Notation	*Value*
Optimal Solution	$\sum_{e\in L} f_e/\tau_e^*$	85.68
Solution by HCCA	$\sum_{e\in L} f_e/\bar{\tau}_e$	86.03
Solution by ACA	$\sum_{e\in L} f_e/\hat{\tau}_e$	92.64
Upper Bound	$\sum_{i\in N} OPT_i$	138.36
2*Optimal Solution	$2\sum_{e\in L} f_e/\tau_e^*$	171.36

We note that in some cases the first two inequalities in (25) as well as the last one hold with equality. For example, in the simple scatternet presented in Fig. 8–A, $\tau_e^* = \bar{\tau}_e = \hat{\tau}_e = 0.5$, $\forall e$ and $\tau_e^i = 0.5$, $\forall i$, $\forall e$. Therefore:

$$8 \;=\; \sum_{e\in L} \frac{f_e}{\tau_e^*} = \sum_{e\in L} \frac{f_e}{\bar{\tau}_e} = \sum_{e\in L} \frac{f_e}{\hat{\tau}_e} < \sum_{i\in N} OPT_i = 2\sum_{e\in L} \frac{f_e}{\tau_e^*} = 16\,.$$

The example described in Fig. 8–B illustrates a different case. In this example $\tau_e^* = \bar{\tau}_e = \hat{\tau}_e = 1/7$, $\forall e$. On the other hand, $\tau_e^1 = 1/7$, $\forall e$ and $\tau_e^i = 1$, $\forall i \neq 1$, $\forall e$. Thus, when ϵ is close enough to $1/7$, the approximate solution is relatively close to the upper bound ($\sum_{i\in N} OPT_i$). Namely:

$$\lim_{\epsilon\to 1/7} \frac{\sum_{i\in N} OPT_i}{\sum_{e\in L} \frac{f_e}{\bar{\tau}_e}} = \lim_{\epsilon\to 1/7} 1 + \frac{1/7-\epsilon}{1-\epsilon} = 1\,.$$

However, in this case the upper bound significantly differs from twice the optimal solution. Namely:

$$\lim_{\epsilon \to 1/7} \frac{2 \sum_{e \in L} \frac{f_e}{\tau_e^*}}{\sum_{i \in N} OPT_i} = 2 \,.$$

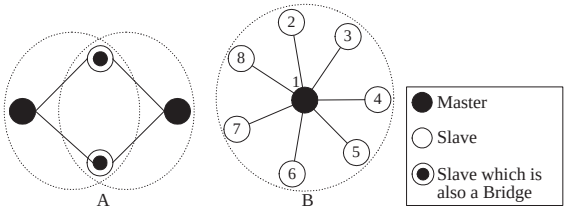

Figure 8: Simple scatternets with given flow rates: (A) $f_e = 1/3 \, \forall e$, (B) $f_e = \epsilon \, \forall e$.

Fig. 9–A illustrates a scatternet with different values of flow represented in terms of the variable x. Fig. 9–B presents the values of the optimal and approximate solutions as well as the upper bound for different values of x. It can be seen that for all flow values, the approximate solutions are very close to the optimal solution and that there is a relatively large difference between $\sum_{i \in N} OPT_i$ and the approximate solutions. Fig. 10–A illustrates a more complex scatternet based on the topology described in [28, Fig. 1]. The corresponding solutions and the upper bound are presented in Fig. 10–B. The results presented in this figure resemble the results presented in Fig. 9–B. We note that in all the cases we have checked, the ratio of the solution obtained by Algorithm HCCA to the optimal solution was much lower than 2.

Finally, it has been mentioned in Section 2.2 that if we wish to take into account the switching overhead

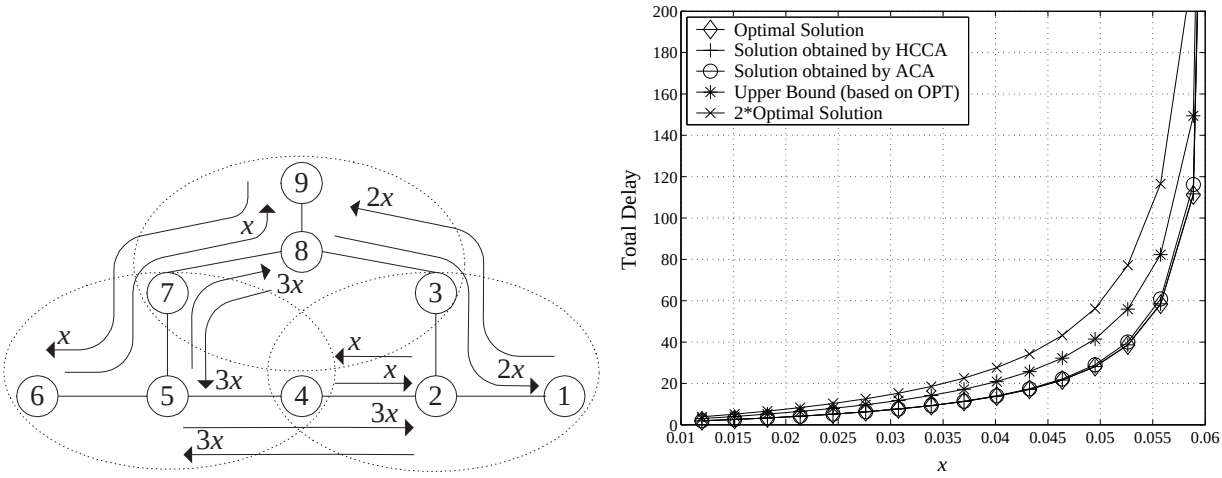

Figure 9: A scatternet: (A) flows expressed in terms of x (an arrow denotes flow along a path), (B) upper bound ($\sum_{i \in N} OPT_i$), optimal solution, and approximate solutions (obtained by Algorithms HCCA and ACA).

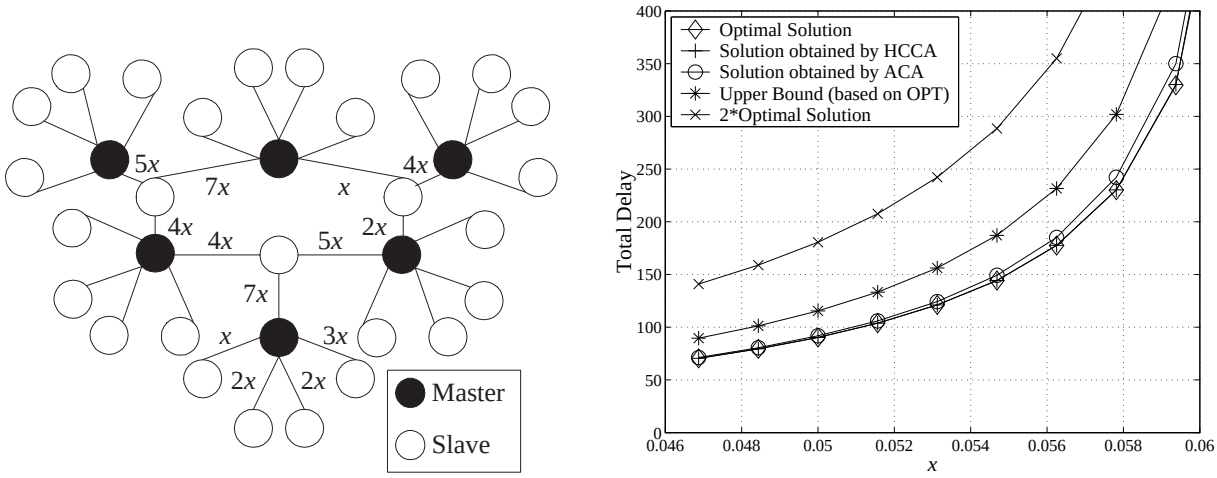

Figure 10: A scatternet: (A) flows expressed in terms of x (the flow values to the non-bridge slaves in every piconet are identical to the values in the lowest piconet), (B) upper bound, optimal solution, and approximate solutions.

incurred by bridges in Bluetooth scatternets, the slack capacity (s_i) of bridge nodes should be defined as $b - \sum_{e \in I(i)} c_e$, where b is the normalized capacity available to the links connected to a bridge. Actually, every node in the network can have a different value of normalized capacity. This property can be used not only to model the overhead incurred by bridges but also to model nodes with different capabilities or different channel conditions. It can be shown that the performance guarantees provided in this paper still hold in this case.

We now present numerical results regarding such a network. Consider the scatternet illustrated in Fig. 9–A. Table 2 presents two different sets of node capacities for this scatternet. Figures 11 and 12 present the corresponding solutions and the upper bound for these two scenarios. It can be seen that although the nodes have different capacities, the approximation ratio is much lower than 2. We note that in most of the cases that we have checked, Algorithm HCCA performed even better than in the situation in which all the nodes have the same capacity.

Table 2: Two sets of node capacities in the scatternet illustrated in Fig. 9–A.

Node	1	2	3	4	5	6	7	8	9
Capacity	1	1	0.7	0.6	1	1	0.7	0.8	1
Capacity	1	0.9	0.5	0.5	0.9	1	0.5	0.9	1

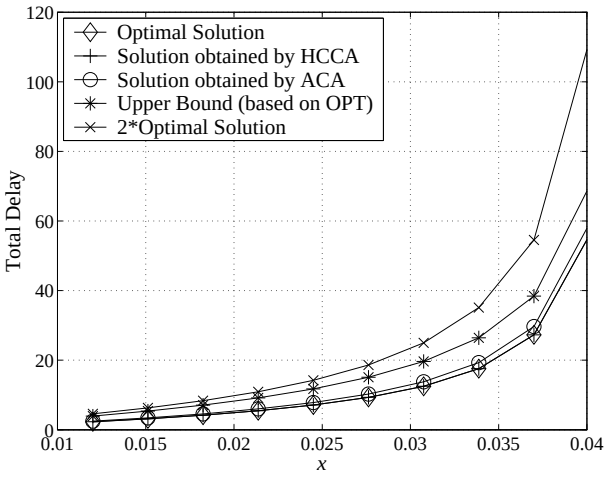

Figure 11: Upper bound, optimal solution, and approximate solutions in the Scatternet presented in Fig. 9–A with the capacities presented in the *first* line of Table 2.

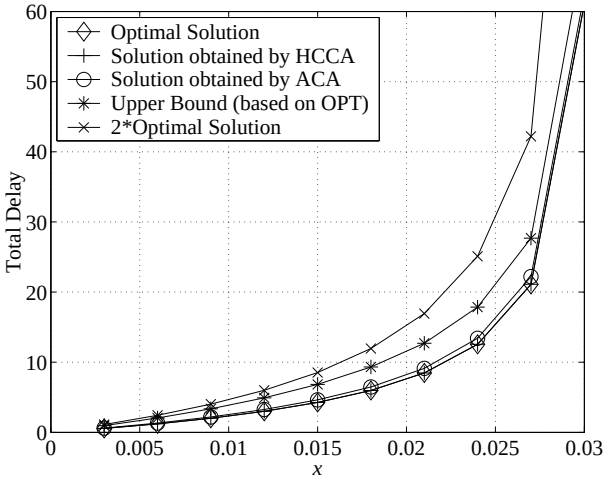

Figure 12: Upper bound, optimal solution, and approximate solutions in the Scatternet presented in Fig. 9–A with the capacities presented in the *second* line of Table 2.

8 Conclusions and Future Study

This paper improves and analyzes heuristic centralized and distributed capacity assignment algorithms and provides an upper bound on their performance. Those algorithms have been designed for Bluetooth scatternets but can be applied to any ad hoc network in which a node transmits to a single neighbor at a time, and in which multiple transmissions can take place as long as they do not share a common node.

We have shown that in bipartite graphs, a centralized solution can be obtained in polynomial time. Yet, due to the need for low-complexity distributed algorithms, we have focused on approximate solutions. First,

we have extended the heuristic algorithms presented in [29] in order to allow them to deal with general decreasing convex objective functions. Then, we have defined a simple approximation algorithm and shown that the ratio between the results obtained by this algorithm and the optimal results is less than two. It has been shown that the heuristic algorithms obtain results which at the worst case are the same as the results obtained by the new algorithm, thus establishing that these algorithms are β-approximation ($\beta < 2$) algorithms for the capacity assignment problem. Moreover, we have shown that although the distributed and centralized algorithms allocate capacity in a different manner, both algorithms converge to the same results. Finally, we have presented numerical results and have compared the approximate solutions to the optimal solution and the upper bound.

There are still many open problems to deal with. For example, it seems that the ratio between the approximate and the optimal solutions is much lower than 2. However, proving this property requires further research. In addition, from a theoretical point of view, it would be interesting to design and analyze algorithms for nonbipartite graphs. On the other hand, from a practical point of view, future study will focus on improving the distributed algorithm and on investigating its performance in a dynamic topology. Finally, we note that a major future research direction is the development of bandwidth allocation methods that will be able to deal with various quality-of-service requirements and to interact with topology construction, scheduling, and routing protocols.

Acknowledgments

This research was supported by the THE ISRAEL SCIENCE FOUNDATION Grant No. 148/03. The research of Gil Zussman was supported by a Marie Curie International Fellowship within the 6th European Community Framework Programme.

We thank the anonymous reviewers for their helpful comments.

Appendix

Proof of Lemma 3: A node i enters the token transfer state due to two possible events:

- Capacity allocation by the node (Step 6 in Fig. 5). In this case it is obvious that following the capacity allocation the node is fully allocated.
- Receipt of the token from a neighbor j that popped its details from the stack (Step 9 in Fig. 6). Neighbor j pops the details of its parent from the stack only if all j's neighbors (including i) are fully allocated.

Proof of Lemma 4: Denote by i the node which initiates the algorithm. Assume that there exists a node j that has been in the allocation state and has not been in the token transfer state.

- If $i = j$, then i will not execute Step 6 (described in Fig. 6), which is a contradiction to the fact that the algorithm halts.
- If j received the token from i, then j will not pop the details of i from the stack (Step 9 in Fig. 6) and therefore, the algorithm will not halt, which is a contradiction.
- Assume that the token traversed the following path $i, k_1, k_2, \ldots, k_l, j$. Node j will not pop the details of k_l from the stack (Step 9 in Fig. 6). Thus, k_l will not pop the details of k_{l-1} and, for similar reasons, k_1 will not pop the details of i. Accordingly, the algorithm will not halt, which is a contradiction.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows*, Prentice Hall Inc., New Jersey, 1993.

[2] S. Baatz, C. Bieschke, M. Frank, C. Kühl, P. Martini, and C. Scholz, "Building Efficient Bluetooth Scatternet Topologies from 1-Factors", *Proc. IASTED WOC 2002*, July 2002.

[3] 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.

[4] S. Basagni, R. Bruno, G. Mambrini, and C. Petrioli, "Comparative Performance Evaluation of Scatternet Formation Protocols for Networks of Bluetooth Devices", *ACM/Kluwer Wireless Networks*, Vol. 10, No. 2, pp. 197-213, Mar. 2004.

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

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

[7] P. Bhagwat and S. P. Rao, "On the Characterization of Bluetooth Scatternet Topologies", *Submitted for Publication*, Available at http://www.winlab.rutgers.edu/˜pravin/publications/papers/bt-top.ps, Dec. 2004.

[8] Bluetooth Special Interest Group, *Specification of the Bluetooth System – Version 1.2*, Nov. 2003.

[9] S. A. Borbash and A. Ephremides, "Wireless Link Scheduling with Power Control", *Proc. WiOpt'04*, Mar. 2004

[10] J. Edmonds, "Maximum Matching and a Polyhedron with (0,1) Vertices", *J. of Research of the National Bureau of Standards*, Vol. 69B, pp. 125–130, 1965.

[11] B. Hajek and G. Sasaki, "Link Scheduling in Polynomial Time", *IEEE Trans. on Information Theory*, Vol. 34, pp. 910–917, Sep. 1988.

[12] L. Har-Shai, R. Kofman, A. Segall, and G. Zussman, "Load Adaptive Inter-Piconet Scheduling in Small-Scale Bluetooth Scatternets", *IEEE Communications*, Vol. 42, No. 7, pp. 136-142, July 2004.

[13] D. S. Hochbaum, "Polynomial and Strongly Polynomial Algorithms for Convex Network Optimization, in Network Optimization Problems", in *Network Optimization Problems (eds: D. Z. Du and P. M. Pardalos)*, pp. 63-92, World Scientific, Singapore, 1993.

[14] D. S. Hochbaum and J. G. Shanthikumar, "Convex Separable Optimization is Not Much Harder than Linear Optimization", *J. of the ACM*, Vol. 37, No. 4, pp. 843-862, Oct. 1990.

[15] IEEE std 802.15.3-2003, "Part 15.3: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specification for High Rate Wireless Personal Area Networks", Sep. 2003.

[16] A. V. Karzanov and S. T. McCormick, "Polynomial Methods for Separable Convex Optimization in Unimodular Linear Spaces with Applications", *SIAM J. Comput.*, Vol. 26, No. 4, pp. 1245-1275, Aug. 1997.

[17] F. P. Kelly, "Charging and Rate Control for Elastic Traffic", *European Trans. on Telecommunications*, Vol. 8, pp. 33-37, 1997.

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

[19] F. Legrand, I. Bucaille, S. Hetuin, L. De Nardis, G. Giancola, M.-G. Di Benedetto, L. Blazevic, and P. Rouzet, "U.C.A.N.'s Ultra Wide Band System: MAC and Routing protocols", *Proc. IWUWBS'03*, June 2003.

[20] M. Minoux, "Solving Integer Minimum Cost Flows with Separable Cost Objective Polynomially", *Mathematical Programming Study*, Vol. 26, pp. 237-239, 1986.

[21] D. Miorandi and A. Zanella, "Performance Analysis of Limited-1 Polling in a Bluetooth Piconet", *Proc. IEEE PIMRC'04*, Sep. 2004.

[22] D. Porcino and W. Hirt, "Ultra-Wideband Radio Technology: Potential and Challenges Ahead", *IEEE Communications*, Vol. 41, No. 7, pp. 66-74, July 2003.

[23] S. Ramanathan, "A Unified Framework and Algorithm for Channel Assignment in Wireless Networks", *ACM/Kluwer Wireless Networks*, Vol. 5, No. 2, pp. 81-94, Mar. 1999.

[24] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed Topology Construction of Bluetooth Personal Area Networks", *Proc. IEEE INFOCOM'01*, Apr. 2001.

[25] S. Sarkar and L. Tassiulas, "End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks", *Proc. IEEE CDC'03*, Dec. 2003.

[26] L. Tassiulas and S. Sarkar, "Maxmin Fair Scheduling in Wireless Networks", *Proc. IEEE INFOCOM'02*, June 2002.

[27] R. M. Whitaker, L. Hodge, and I. Chlamtac, "Bluetooth Scatternet Formation: a Survey", To appear in *Ad Hoc Networks*, Vol. 3, 2005.

[28] W. Zhang and G. Cao, "A Flexible Scatternet-wide Scheduling Algorithm for Bluetooth Networks", *Proc. IEEE IPCCC'02*, Apr. 2002.

[29] G. Zussman and A. Segall, "Capacity Assignment in Bluetooth Scatternets - Optimal and Heuristic Algorithms", *ACM/Kluwer Mobile Networks and Applications (MONET)*, Vol. 9, No. 1, pp. 49-61, Feb. 2004.

[30] G. Zussman, A. Segall, and U. Yechiali, "Bluetooth Time Division Duplex - Analysis as a Polling System", *Proc. IEEE SECON'04*, Oct. 2004.

[31] G. Zussman, U. Yechiali, and A. Segall, "Exact Probablistic Analysis of the Limited Scheduling Algorithm for Symetrical Bluetooth Piconets", *Proc. IFIP-TC6 PWC'03, LNCS Vol. 2775 (eds: M. Conti et al.)*, Springer, Sep. 2003.