Performance issues of Bluetooth scatternets and other asynchronous TDMA ad hoc networks

Theodoros Salonidis and Leandros Tassiulas
Electrical and Computer Engineering Department and Institute of Systems Research
University of Maryland, College Park
{thsalon,leandros}@eng.umd.edu

***Abstract*—**
A common assumption of TDMA-based wireless ad hoc networks is the existence of network-wide slot synchronization. In practice such a mechanism is difficult to support. In asynchronous TDMA systems each link uses a local time slot reference provided by the hardware clock tick of one of the node endpoints. Inevitably, slots will be wasted when nodes switch time slot references. This restricts the rate allocations that can be supported when compared to a perfectly synchronized system. We address this practical performance issue for the case of Bluetooth, a wireless technology operating according to the asynchronous TDMA communication paradigm. We introduce scheduling algorithms that not only guarantee upper bounds on the generated overhead but also target its minimization.

I. Introduction

A wireless ad hoc network is a collection of nodes equipped with radio interfaces forming a multi-hop all-wireless infrastructure. Time division multiple access (TDMA) is a well-known medium access scheme for deterministic bandwidth allocation and quality of service (QoS) provision in ad hoc networks. According to TDMA, bandwidth can be allocated to the network links using a schedule of period T_{system} slots. At every slot of such a schedule, several links are activated for transmission such that no conflicts occur at the intended receivers. The amount of conflict-free slots a link receives within a period determines its allocated bandwidth.

A central performance issue that arises in a TDMA-based ad hoc network is determination of the set of allocations it can achieve. Given a set of end-to-end sessions, the sum of their requested rates over the links they traverse creates a demand allocation for each link. A demand link rate allocation $\boldsymbol{r} = [r_l]$ ($0 \leq r_l \leq 1$) is feasible if the network can allocate $\tau_l = \lfloor r_l \cdot T_{system} \rfloor$ conflict-free slots to every link l without exceeding the system period. Determination of feasibility is intrinsically coupled with an optimization problem: to find a link schedule of minimum period that realizes slot allocation $\boldsymbol{\tau} = [\tau_l]$. If the solution of this problem is less than the system period, then the allocation is feasible.

Optimal link scheduling in wireless ad hoc networks has been studied by [2] [3] for various interference constraints. These studies, along with most proposed centralized or distributed TDMA protocols for slotted ad hoc networks, assume the time slot boundaries are provided by a global system clock. This system-wide synchronization mechanism is not always possible to achieve in the distributed ad hoc network setting. Bluetooth [1] is a new TDMA wireless technology that enables the formation of ad hoc networks called scatternets. While being a slotted system, Bluetooth has the interesting feature of not supporting a global slot synchronization mechanism. Instead, time reference is provided locally for each link by one of the node endpoints acting as master. Inevitably, slots will be wasted when nodes switch time slot references as slaves. This phenomenon has been reported in works related to scatternet scheduling [8] [9] [10] [11] [12] as a source of overhead. However, no formal study has examined its effect on the ability of the system to allocate bandwidth. This ability is linked to the determination of the feasible allocations region, or, equivalently, the solution of the related link schedule optimization problem.

Given a demand allocation, the minimum period achieved by an asynchronous TDMA system is expected to be greater than the one achieved by a perfectly synchronized one. This is because the various time reference switches over time can have a cumulative additive effect on the overall minimum period required by the asynchronous system. The increase in the minimum period is essentially the overhead introduced by system asynchronicity.

Based on this observation, we can use a two-step approach to address the link schedule optimization problem for the asynchronous scatternet setting. The first step finds a synchronized schedule of minimum period that realizes the demand allocation. Bluetooth falls in the category of multi-channel systems studied in [3] and the algorithms contained therein can be used for this purpose. The second step, our contribution, utilizes the optimal synchronized schedule to find an asynchronous schedule of minimum overhead.

It turns out that the overhead depends on the order of link activations in the reference synchronized schedule. We introduce two algorithms for addressing this problem. The first algorithm derives a minimum overhead asynchronous schedule for a specific link activation ordering of the synchronized schedule. It also has an upper bound for the overhead it generates for any possible input ordering or scatternet configuration. Using this algorithm it is possible to reach the optimal solution by executing it over all possible orderings. This leads to a problem of combinatorial nature that prohibits exhaustive search for large problem sizes. To this end we introduce a heuristic algorithm of polynomial complexity. The heuristic is shown to have excellent performance for problem sizes where the optimal can be computed. For large problem sizes we investigate the effect

of the various system parameters to the generated overhead and use the derived upper bound as a performance measure.

The paper is organized as follows. Section II is an introduction to the architecture of Bluetooth scatternets and related work on their scheduling. Section III introduces a scheduling framework for allocating bandwidth in the asynchronous scatternet setting by means of periodic conflict-free link schedules. Sections IV and V provide the formulation of the asynchronicity overhead problem and the algorithms used for minimizing it. Section VI evaluates the algorithms' performance. Section VII concludes the paper.

II. Bluetooth architecture

Every Bluetooth node has an internal "native" system clock that determines the timing of the radio transceiver. Native clocks of different nodes are not synchronized. Synchronization is locally acquired when nodes are grouped in distinct communication channels called "piconets". Each channel is defined by a frequency hopping sequence derived from the identity of one of the nodes acting as "master". The master provides its native clock as the piconet time slot reference. Each slot supports full-duplex communication initiated by the master: During the first part of the slot the master polls a slave; during the second part a slave responds if polled by the master.

Bluetooth imposes the maximum number of piconet members to be eight. However, piconets can be interconnected via bridge nodes to form a larger ad hoc network known as a "scatternet". Bridges can timeshare between multiple piconets, receiving data from one piconet and forwarding it to another. A bridge may act as master in a single piconet and slave in others (termed as M/S bridge) or act as slave in multiple piconets (termed as S/S bridge).

The Bluetooth technology standard [1] has not yet specified the way bridges should schedule their visits in different piconets; this is currently a subject of intense research effort. Emphasis is placed on distributed scheduling schemes and approaches can be categorized according to the degree of coordination they offer. According to "hard coordination" schemes [12][7], link scheduling is performed in such a way that when a master polls a slave, this slave is guaranteed to be tuned on this piconet. Since no transmission conflicts exist, these schemes can potentially achieve strict bandwidth allocation guarantees. However, there exists an associated implementation and communication complexity for maintaining the conflict-free property especially when the scatternet becomes highly dynamic. Soft coordination schemes [10][9][11] tradeoff perfectly conflict-free transmissions for lower complexity. The downside here is that this comes to a loss of the ability to provide bandwidth guarantees.

While there is still a simplicity v.s. performance debate between the two approaches, bandwidth loss due to piconet switching always exists when slaves switch piconet time references. In the next section we introduce a hard coordination scheduling framework for overhead minimization. There are mainly two reasons for doing this. First, in this case the overhead is naturally linked to the ability of the system to allocate bandwidth. Second, conflict-free scheduling is the best we can do to minimize the overhead and this provides a useful point of reference.

III. Scatternet communication model

The scatternet is represented as a directed graph $G(N, E)$. A directed edge $(i, j) \in E$ signifies that nodes i and j are within wireless range and they have established a Bluetooth link where i is the master and j the slave.

We assume that transmissions on a piconet are cleanly received by a node listening on that piconet despite any in-range transmissions that may be happening at other piconets. Recent studies [4] have indicated that this is a good approximation for the frequency hopping sequences used in Bluetooth. We also assume no losses due to channel errors. The access problem arises because each Bluetooth node has a single radio transceiver and can communicate (transmit or receive) to at most one piconet at a time. Thus, nodes need to coordinate their presence on links during mutual time intervals.

Based on its own hardware clock, a node i divides time in fixed-sized slots and coordinates transmissions on its adjacent links using a local link schedule $\boldsymbol{S}_i$ of period T_{system} slots. The local schedule determines communication action for the duration of a slot: the node can either be active on a single link (polling if master or responding to a poll if slave) or remain idle. Because the local schedules are not synchronized, for conflict-free communication on τ_l consecutive slots on link l, the master must allocate τ_l slots in its local schedule for polling, while the slave must allocate at least $\tau_l + 1$ time-overlapping slots for tuning to the frequency hopping sequence and aligning to the time reference of this master. Thus, certain slots in a node's local schedule are wasted. More specifically, an extra slot is needed each time a node switches to a new piconet acting as slave.

A *slot allocation* $\boldsymbol{\tau}$=[τ_l] is the number of slots every link l transmits conflict-free during T_{system} slots and equals the number of slots allocated to the local periodic schedule of the link master. Given a slot allocation, the various piconet switches over time have a cumulative effect on the overall minimum period required by the asynchronous system. Figure 1 illustrates this phenomenon: under perfect synchronization, an allocation of 3 slots per link could be realized by a minimum period of 6 slots. Both asynchronous schedules (a) and (b) need a larger period for realizing it. In addition, the amount of overhead depends on the order links are activated in the schedule. In (a) only two slots are wasted, while in (b), node B switches time reference every other slot, yielding a higher period of 12 slots.

According to this example, the *asynchronicity overhead* for realizing a demand slot allocation can be defined as an increase in period with respect to a perfectly synchronized system. Also, the amount of overhead depends on the link activation order. The problem then is to find an asynchronous schedule and link activation order of minimum overhead. The following sections provide with a formulation of this problem and an approach for solving it.

IV. Equivalent schedules

A link activation set consists of links that can simultaneously transmit without conflicts to the intended receivers. A synchro-

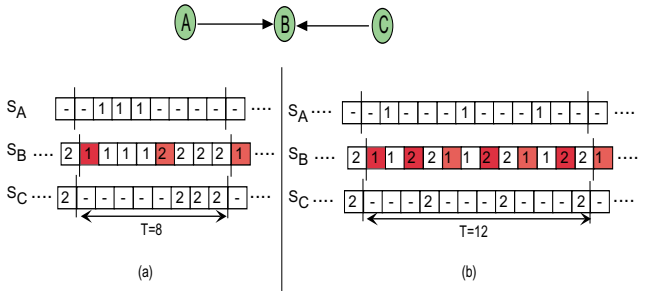


Fig. 1. Node B acts as slave on both links. The two asynchronous schedules realize slot allocation $(3, 3)$ with different periods, depending on the link activation order.

nized link schedule $\tilde{\boldsymbol{S}}$ of period $\tilde{T}$ is a collection of link activation sets $\{A_k : 1 \leq k \leq \tilde{T}\}$. A **synchronized schedule instance** $\tilde{S}^{(\pi)}$ is a periodic sequence of a specific ordering π of the link activation sets of $\tilde{\boldsymbol{S}}$:

$$\tilde{S}^{(\pi)} = (A_{\pi(1)}, ..., A_{\pi(\tilde{T})}). \quad (1)$$

where π is a mapping of the indexes $\{1, ..., \tilde{T}\} \rightarrow \{1, ..., \tilde{T}\}$. Given a reference synchronized schedule instance $\tilde{S}^{(\pi)}$, an **equivalent** asynchronous schedule $\boldsymbol{S}^{(\pi)}$ is the one that yields minimum overhead for this order of link activations.

Algorithm EQUIVALENT constructs $\boldsymbol{S}^{(\pi)}$ incrementally by iterating over the link activation sets of $\tilde{\boldsymbol{S}}^{(\pi)}$. During iteration k, let l be a link in activation set $A_{\pi(k)}$ and i and j be its master and slave endpoints. Also let $p_i^{(k-1)}$ and $p_j^{(k-1)}$ be the last assigned slot positions in the local schedules $\boldsymbol{S}_i^{(\pi)}$ and $\boldsymbol{S}_j^{(\pi)}$ respectively.

First, master i determines slot $p_i^{(k)}$ to be assigned to link l in $\boldsymbol{S}_i^{(\pi)}$. If l was not activated in the previous iteration and $p_j^{(k-1)} \geq p_i^{(k-1)}$, then $p_i^{(k)}$ is the earliest unassigned slot whose start time exceeds the end time of slot $p_j^{(k-1)}$ in $\boldsymbol{S}_j^{(\pi)}$. Otherwise, $p_i^{(k)} = p_i^{(k-1)} + 1$. Any intermediate slots between $p_i^{(k-1)}$ and $p_i^{(k)}$ are assigned idle in $\boldsymbol{S}_i^{(\pi)}$.

Then, slave j determines $p_j^{(k)}$ as the earliest unassigned slot in $\boldsymbol{S}_j^{(\pi)}$ whose end time exceeds the end time of $p_i^{(k)}$ in $\boldsymbol{S}_i^{(\pi)}$. Any intermediate slots between $p_j^{(k-1)}$ and $p_j^{(k)}$ are assigned to link l in $\boldsymbol{S}_j^{(\pi)}$.

The same steps are performed for every link l in $A_{\pi(k)}$. For every node n not considered in iteration k, $p_n^{(k)} = p_n^{(k-1)}$. At the end of iteration k, the **forward progress** $f(k)$ is the maximum progress over all local schedules after this iteration:

$$f(k) = \max_{n \in N} \{p_n^{(k)}\} \quad (2)$$

After $\tilde{T}$ iterations, the asynchronous schedule period $T^{(\pi)}$ is set to the forward progress $f(\tilde{T})$. Then, the algorithm restarts from $A_{\pi(1)}$ and performs one or more extra iterations until all nodes assign their local schedules up to slot $T^{(\pi)}$. Upon termination, all nodes use the first $T^{(\pi)}$ slots in their local schedules to form an asynchronous schedule with this period. The algorithm operation is illustrated in the example of Figure 2.

EQUIVALENT has the following important properties (established in [13]):

1) The resulting asynchronous schedule incurs the minimum possible overhead for the link activation ordering corresponding to $\tilde{\boldsymbol{S}}^{(\pi)}$.
2) If $\tilde{T}$ is the period of $\tilde{\boldsymbol{S}}^{(\pi)}$, the period $T^{(\pi)}$ of the resulting asynchronous schedule is always upper bounded by $2\tilde{T}$.

Property 2 states that the maximum possible overhead of an equivalent asynchronous schedule is $\tilde{T}$ slots. This leads to the following statement for feasibility of allocations in scatternets:

Corollary on feasibility: Consider a demand allocation $\boldsymbol{\tau}$ and a scatternet operating with a period T_{system}. If $\boldsymbol{\tau}$ can be realized by a *synchronized* schedule $\tilde{\boldsymbol{S}}$ of minimum period $\tilde{T}(\boldsymbol{\tau}) \leq \lfloor T_{system}/2 \rfloor$, then $\boldsymbol{\tau}$ is guaranteed to be feasible by the scatternet.

The corollary (also proved in [13]) establishes that EQUIVALENT can realize at least half the allocations that are feasible under perfect synchronization. Also for allocations $\boldsymbol{\tau}$ for which the condition $\tilde{T}(\boldsymbol{\tau}) \leq \lfloor T_{system}/2 \rfloor$ holds, any reference synchronized schedule instance can be used to generate an asynchronous schedule realizing this allocation. If the condition does not hold we must solve the optimization problem addressed in the next section.

V. Minimum-period asynchronous schedules

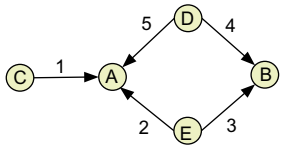
A. Optimal algorithm

The optimal asynchronous schedule can be found by executing EQUIVALENT for all $\tilde{T}!$ synchronized schedule instances $\tilde{\boldsymbol{S}}^{(\pi)}$ and selecting the minimum period equivalent schedule $\boldsymbol{S}^{(\pi)}$. However, exhaustive search is prohibitive even for small values of $\tilde{T}$.

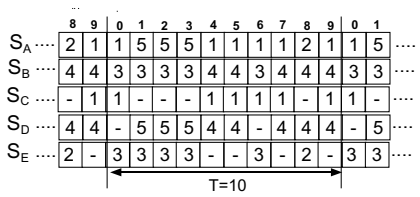
A link activation set may appear multiple times in the reference synchronized schedule. The search space can be reduced if we only consider reference schedules where all instances of each link activation set are scheduled in consecutive slots. This is because there are no switching slots generated by EQUIVALENT when $A_{\pi(k-1)} = A_{\pi(k)}$ and the overhead is zero during this iteration. If $M(\tilde{\boldsymbol{S}})$ is the set of *distinct* link activation sets appearing in the reference schedule, we only need to search $|M(\tilde{\boldsymbol{S}})|!$ schedule instances instead of $\tilde{T}!$. Unfortunately even $|M(\tilde{\boldsymbol{S}})|$ can be prohibitively large for exhaustive searches. In this case, we resort to the heuristic algorithm introduced in the next section.

B. MIN_PROGRESS

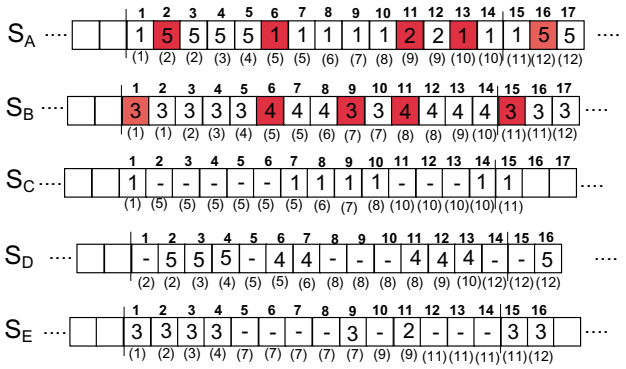
MIN_PROGRESS is a heuristic for overhead minimization that consists of two phases. The first phase determines an ordering π_h of the distinct link activation sets in $M(\tilde{\boldsymbol{S}})$. The second phase first forms a synchronized schedule instance where distinct link activation sets are ordered according to π_h and the instances of each set are activated in consecutive slots. This



(a) Scatternet configuration: Nodes C,D,E are masters while A,B are slaves to all their adjacent links.



(b) Reference synchronized schedule of (minimum) period $\tilde{T} = 10$, realizing allocation $(\tau_1, \tau_2, \tau_3, \tau_4, \tau_5) = (6, 1, 5, 5, 3)$



(c) The numbers in parentheses indicate the iteration where the slot was placed by the algorithm on each node's local schedule. Switching slots are indicated by red. The equivalent schedule period is determined at the 10^{th} iteration and is equal to 14. Two additional iterations are performed so that all nodes fill their local schedules up to this period.

(k)	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
f(k)	0	2	3	4	5	7	8	10	12	13	14
$p_A^{(k)}$	0	1	3	4	5	7	8	9	10	12	14
$p_B^{(k)}$	0	2	3	4	5	7	8	10	12	13	14
$p_C^{(k)}$	0	1	1	1	1	7	8	9	10	10	14
$p_D^{(k)}$	0	0	2	3	4	6	7	7	11	12	13
$p_E^{(k)}$	0	1	2	3	4	4	4	9	9	11	11

(d) Evolution of the $p_n^{(k)}$ and progress $f(k)$.

Fig. 2. An example of the EQUIVALENT algorithm execution

synchronized schedule instance is then input to EQUIVALENT to generate the equivalent asynchronous schedule.

We now describe phase I that selects permutation π_h. An asynchronous schedule is constructed using only the distinct link activation sets instead of all their instances. The sets are added to the asynchronous schedule in the same way as instances are added in EQUIVALENT. Upon initialization, an arbitrary set of $M(\tilde{\boldsymbol{S}})$ is added to the asynchronous schedule. Let $U^{(k-1)}$ be the set of all unassigned link activation sets at the start of iteration k. The addition of each set M^{α} of $U^{(k-1)}$ will generate a forward progress $f(\alpha, k)$ for the asynchronous schedule. The algorithm selects the link activation set yielding minimum forward progress, with ties being broken arbitrarily. Let M^{α_k} be the selected set. Then the k-th entry of π_h is set to α_k. At the end of iteration k, M^{α_k} is removed from the U-set ($(U^k = U^{(k-1)} - \{M^{\alpha_k}\})$). The same steps are repeated until the U-set becomes empty after $|M(\tilde{\boldsymbol{S}})|$ iterations.

The complexity of MIN_PROGRESS is $O(|M(\tilde{\boldsymbol{S}})|^2)$ and is dominated by phase I: During iteration k, $|M(\tilde{\boldsymbol{S}})| - k$ sets are considered for addition in the asynchronous schedule. The total number of link activation sets considered during phase I is $(|M(\tilde{\boldsymbol{S}})| - 1) + (|M(\tilde{\boldsymbol{S}})| - 2) + ... + 1 = |M(\tilde{\boldsymbol{S}})|(|M(\tilde{\boldsymbol{S}})| - 1)/2 = O(|M(\tilde{\boldsymbol{S}})|^2)$.

VI. Performance Evaluation

A. Experimental setting

The performance of $MIN\_PROGRESS$ must be evaluated over a variety of scatternet topologies and optimal reference synchronized schedules. For arbitrary topologies the problem of determining a minimum period synchronized schedule for a link demand allocation $\boldsymbol{\tau}$ is NP-complete [6]. If the topology is bipartite, the minimum period $\tilde{T}(\boldsymbol{\tau})$ is equal to the maximum node utilization imposed by $\boldsymbol{\tau}$:

$$\tilde{T}(\boldsymbol{\tau}) = \max_{i \in N} \sum_{l \in L(i)} \tau_l. \tag{3}$$

where $L(i)$ is the set of adjacent links to node i. Thus for bipartite topologies we can easily construct optimal reference synchronized schedules of period $\tilde{T}$ for arbitrary allocations: we generate an arbitrary conflict-free schedule of period $\tilde{T}$, where at least one node transmits during all $\tilde{T}$ slots on its adjacent links.

We consider $|N|$-node bipartite topologies[1] with $|N|/2$ nodes per bipartite set. This provides a baseline topology of $|N|^2/4$ links. We use the restrictive parameters B_{max} and f to generate various topologies from the baseline. The piconet degree parameter B_{max} is an upper bound on the number of piconets a node can participate. Such a constraint would arise in practice to avoid excessive overhead due to piconet switching. In addition, Bluetooth restricts the number of links where a node can act as master to 7. Combined with B_{max}, this provides an upper bound of $B_{max} + 6$ to the overall link degree of each node in the topologies we consider. The density parameter f ($0 \leq f \leq 1$) generates topologies where an arbitrary

[1]Bipartite topologies arise very frequently in the Bluetooth setting. For example, a scatternet where only S/S bridges exist (i.e. nodes acting only as slaves on their adjacent links) is by definition bipartite.

$f \times 100\%$ links of the baseline topology remain intact while the rest have been removed.

Given a topology constructed as above, asynchronicity is introduced by master-slave role assignments on the links and introduction of arbitrary phase differences on the hardware clocks of the nodes in the network.

B. Performance of MIN_PROGRESS with respect to optimal

Six 20-node bipartite topologies (10 masters and 10 S/S bridges) of variable density B_{max} are considered in this experiment. For each topology we randomly generate 100 reference synchronized schedules of period $\tilde{T} = 7$. This period allows exhaustive search and determination of the optimal asynchronous schedule. Figure 3 compares the resulting optimal and MIN_PROGRESS periods averaged over all reference schedules. In general, MIN_PROGRESS exceeds the optimal by less than one slot on the average, while in topology 5 by 1.3 slots on the average. The optimal and MIN_PROGRESS periods increase with B_{max} and for $B_{max} = 7$ they both come very close to 14 slots, the upper bound of $EQUIVALENT$. This stems from B_{max} being equal to the small reference period $\tilde{T}$: Bridge nodes with such a piconet degree need to switch time reference almost every slot regardless the ordering of link activations.

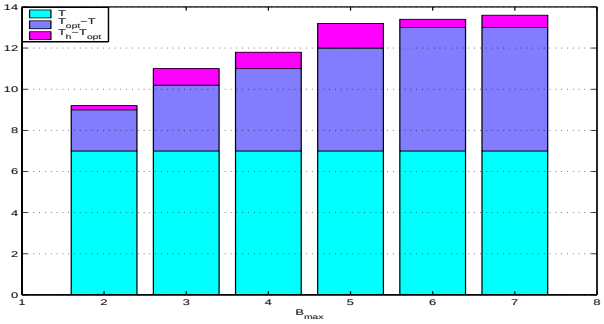


Fig. 3. Each bar graph corresponds to a different 20-node bipartite scatternet configuration, where density increases by varying B_{max} from 2 to 7. The reference synchronized schedule period is 7 slots. The optimal T_{opt} and the heuristic T_h asynchronous periods of each bar are averages of 100 reference synchronized schedules.

C. Performance of MIN_PROGRESS for large problem sizes

1) **Effect of density***:* In this set of experiments, a 100-node (50 masters, 50 S/S bridges) baseline bipartite topology is used. For each set (B_{max}, f) we generate 10 topologies and for each topology, 100 arbitrary reference synchronized schedules of period $\tilde{T}$. The overhead is plotted as the %increase in the reference period $\tilde{T}$. If T_h is the period computed by MIN_PROGRESS, this quantity is $\frac{T_h - \tilde{T}}{\tilde{T}}$. A value of 100% denotes that MIN_PROGRESS yields an overhead equal to the EQUIVALENT upper bound of $2\tilde{T}$.

Figure 4 investigates the effect of B_{max} on the overhead generated by MIN_PROGRESS. For fixed $\tilde{T}$ the overhead consistently increases with B_{max}. At $\tilde{T} = 28$, the overhead is 15% when $B_{max} = 2$ but reaches 60% when $B_{max} = 7$. The overhead decreases as the reference period increases. At $B_{max} = 7$ the overhead reduces to 30% for $\tilde{T} = 896$ slots. While this decrease is more drastic for transitions between smaller periods (e.g. from 28 to 56 slots), it is less for larger periods (e.g. from 448 to 896 slots). This implies that, in general, there may still be a non-negligible overhead even if the system uses a large period. Similar trends arise in Figure 5 where B_max is fixed to 7 and only parameter f is used to vary the topology density. The overhead generally increases with network density regardless enforcement of a particular bound on the piconets each node can participate.

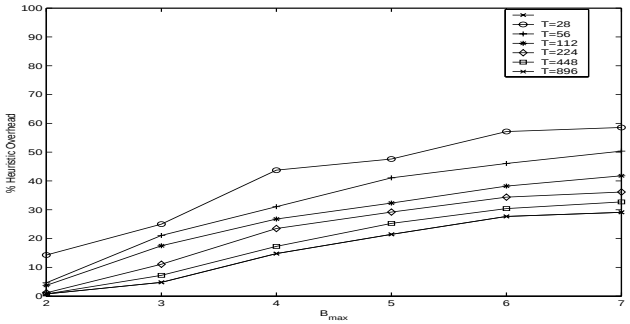


Fig. 4. Overhead of MIN_PROGRESS for 100-node scatternets as B_{max} and $\tilde{T}$ vary. f is set to 1.0.

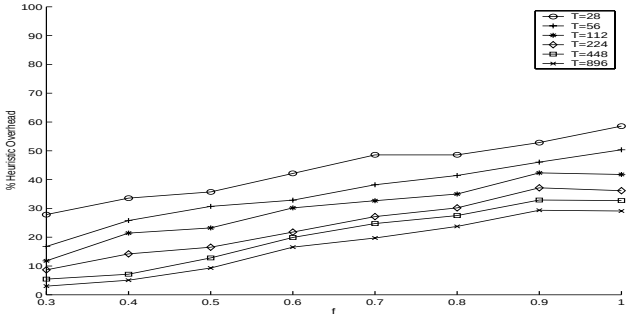


Fig. 5. Overhead of MIN_PROGRESS for 100-node scatternets as B_{max} and $\tilde{T}$ vary. B_{max} is set to 7.

2) **Effect of demand slot allocation***:* The previous experiments investigated the algorithm performance averaged over arbitrary demand allocations and scatternet topologies. A natural question is whether there exists a scatternet role assignment and/or demand allocation for which the generated asynchronicity overhead is maximized. In this section we make a first attempt to informally classify such worst case instances and then test our intuition via simulations.

Let $G(N, E)$ be a bipartite topology graph and $\Psi^{(\tilde{T})}$ the set of all allocations realized by a synchronized schedule of minimum period $\tilde{T}$. For any allocation $\boldsymbol{\tau}$ of $\Psi^{(\tilde{T})}$, let $BN(\boldsymbol{\tau})$ be the set of bottleneck nodes that receive maximum utilization $\tilde{T}$ under $\boldsymbol{\tau}$.

$$BN(\boldsymbol{\tau}) = \{n : arg\max_{i \in N} \sum_{j \in N(i)} \tau_{ij}\}. \quad (4)$$

We conjecture that maximum overhead will be generated if the following conditions hold for a demand allocation $\boldsymbol{\tau}^{max}$ in $\Psi^{(\tilde{T})}$ and at least one of the bottleneck nodes in $BN(\tau^{max})$:

- **P1:** In addition to maximum utilization, the node must have the maximum number of adjacent links in the network.

- **P2:** The node has been assigned as an S/S bridge.
- **P3:** Allocation $\boldsymbol{\tau}^{max}$ is such that the node is requested to allocate an equal number of slots to its adjacent links.

The intuition in the above conditions is that a maximum utilization node will need to be considered at every iteration of an overhead minimization algorithm. Also since this is a node of maximum degree and acts as an S/S bridge it will visit the maximum possible number of piconets in the system (B_{max}). If the requested slots are evenly distributed for this node, then we can show that the overhead will be maximized under the worst activation ordering if its adjacent links. According to [**?**][7], a maxmin fair allocation in a synchronized multi-channel wireless ad hoc network maximizes utilization of the maximum degree nodes in the network. If at least one of these nodes is also assigned as an S/S bridge, then the above conditions hold for at least one node in the network.

Figure 6 compares the MIN_PROGRESS overhead resulting from a maxmin fair synchronized schedule and the average MIN_PROGRESS overhead over 100 other schedules realizing arbitrary allocations. (The maxmin fair schedule is computed using the algorithm in [7]). Each point in the bar graphs is the average of the overheads generated by the scatternet topologies of Figures 4 and 5. Each bar graph corresponds to a different synchronized schedule period $\tilde{T}$. As expected, the average

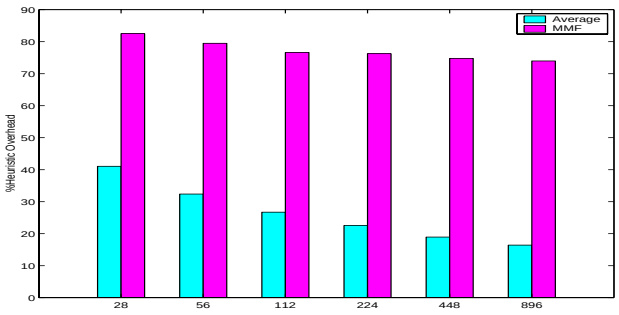


Fig. 6. Comparing the MIN_PROGRESS overhead for maxmin fair allocation with the average MIN_PROGRESS overhead generated by arbitrary allocations. Both quantities are averaged over all topologies considered in Figures 4 and 5

MIN_PROGRESS overhead for arbitrary allocations decreases as the system period increases. However, the one due to the maxmin fair allocation does not change significantly and it is in the order of 80% for all cases. This shows that the overhead can be very high for the allocations we identified even if we use an overhead minimization algorithm such as MIN_PROGRESS. A counter-intuitive result is that the overhead remains constant even if the period $\tilde{T}$ increases. Nevertheless, it will always be less than the upper bound $\tilde{T}$ given by Theorem 2.

VII. Conclusions

In this paper we addressed for the first time the problem of minimizing the piconet switching overhead in Bluetooth scatternets. This overhead arises due to slots wasted when bridge nodes synchronize to the different piconet time references. While the problem was investigated in the Bluetooth context, the results apply to any wireless ad hoc network using slotted TDMA access and multiple local time references instead of a global synchronization mechanism.

It was demonstrated that this overhead can significantly affect the bandwidth allocation ability of a scatternet if no measures are taken to minimize it. We introduced two scheduling algorithms that aim for overhead minimization while ensuring that the generated overhead has an upper bound regardless of the scatternet or demand allocation at hand. The first algorithm reaches the optimal solution but cannot be applied to large problem sizes because it relies on exhaustive search. For large problem sizes a heuristic algorithm was devised and through simulations it was shown to have excellent performance. We also identified certain conditions on demand allocations and scatternet configurations for which the overhead can be high even if an overhead minimization algorithm is run. We outlined the general properties of such allocations and verified our intuition through simulations. A formal study of the exact nature of these allocations is an interesting future work direction.

Both the optimal and heuristic overhead minimization algorithms are centralized and can be used in settings where global information is available. More important though is the fact that that they can provide design insights and be used as a reference performance measure for distributed overhead-aware scatternet scheduling protocols.

Finally, we believe that the derivation of a similar overhead minimization framework for ”soft-coordination” scatternet scheduling schemes is another challenging open research issue.

References

[1] Bluetooth Special Interest Group,*Specification of the Bluetooth system, ver 1.0B.* , www.bluetooth.com, October 2000.

[2] E. Arikan, *Some complexity results about packet radio networks*. IEEE Trans. Inform. Theory, Vol. IT-30 pp. 681-685, July 1984.

[3] B. Hajek and G. Sasaki, *Link Scheduling in Polynomial Time*. IEEE Trans. Inform. Theory, No 5, Vol. 34, 1988.

[4] A. Kumar, R. Gupta *Capacity Evaluation of Frequency Hopping Based Ad-hoc Systems*. Proceedings of ACM SIGMETRICS 2001

[5] M. Post, P. Sarachik and A Kershenbaum, *A Biased Greedy Algorithm for Scheduling Multihop Radio Networks*. 19th Annu. Conf. on Information Sciences and Systems, Johns Hopkins Univ., March 1985.

[6] I. Holyer, *The NP-completeness of edge coloring* SIAM J. Computing 10 (1981), 169-197.

[7] T. Salonidis and L. Tassiulas, *Distributed on-line schedule adaptation for balanced slot allocation in Bluetooth scatternets and other ad hoc network architectures*. Technical Report TR 2002-24, Institute of Systems Research (ISR), University of Maryland, College Park.

[8] G. Miklos et al, *Performance Aspects of Bluetooth Scatternet Formation*. Proceedings of IEEE/ACM MobiHoc, Boston, MA, Aug. 2000.

[9] N.Johansson, F. Alriksson, U. Jonsson, *JUMP mode - a dynamic window-based scheduling framework for Bluetooth scatternets*. Proceedings of IEEE/ACM MobiHoc, Long Beach CA, Oct. 2001.

[10] A. Racz, G. Miklos, F. Kubinszky, A. Valko *A Pseudo Random Coordinated Scheduling algorithm for Bluetooth Scatternets*. Proceedings of IEEE/ACM MobiHoc, Long Beach CA, Oct. 2001.

[11] Simon Baatz, Matthias Frank, Carmen K uhl, Peter Martini, Christoph Scholz, *Bluetooth Scatternets: An Enhanced Adaptive Scheduling Scheme*. Proceedings of Infocom 2002, New York, 2002.

[12] N. Johansson, U. Korner, L. Tassiulas, *A distributed scheduling algorithm for a Bluetooth scatternet*. In Proc. Of the 17th International Teletraffic Congress, ITC ’17. Salvador da Bahia, Brazil, Sep. 2001.

[13] T. Salonidis and L. Tassiulas, *Performance Issues of Bluetooth scatternets and other asynchronous TDMA ad hoc networks*. Technical Report TR 2002-52, Institute of Systems Research (ISR), University of Maryland, College Park.