

Asynchronous TDMA ad hoc networks: Scheduling and Performance*

THEODOROS SALONIDIS [†] AND LEANDROS TASSIULAS ^{†,‡}

[†] Department of Electrical and Computer Engineering, University of Maryland at College Park, MD 20742, USA

[‡] Department of Computer and Communication Engineering, University of Thessaly, 38221 Volos, Greece
{thsalon,leandros}@eng.umd.edu

Abstract. A common assumption of TDMA-based wireless ad hoc networks is the existence of network-wide slot synchronization. Such a mechanism is difficult to support in practice. 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 are wasted when nodes switch time slot references. This restricts the rate allocations that can be supported when compared to a perfectly synchronized system.

To address this practical performance issue we first introduce a general framework for conflict-free scheduling in asynchronous TDMA networks. We then propose scheduling algorithms that target overhead minimization while ensuring upper bounds on the generated overhead. Through extensive simulations the algorithm performances are evaluated in the context of Bluetooth, a wireless technology that operates according to the asynchronous TDMA communication paradigm.

1 INTRODUCTION

An ad hoc network is a collection of nodes equipped with radio interfaces that forms an all-wireless communication infrastructure. Time division multiple access (TDMA) is a well-known medium access method 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. During every slot, several links are activated for transmission such that no conflicts occur at the intended receivers. The number of conflict-free slots each link receives within a system 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 that can be achieved. Consider a set of end-to-end sessions sharing the network. The sum of their requested rates over the links they traverse creates a demand allocation for each link. A link rate allocation $\mathbf{r} = [r_l]$ ($0 \leq r_l \leq 1$) 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 minimum period does not exceed T_{system} slots, then the allocation is feasible.

Optimal link scheduling in wireless ad hoc networks has been investigated for various interference constraints [1, 2]. These studies (along with most proposed centralized or distributed TDMA protocols for slotted ad hoc networks) assume the slot boundaries are provided by a global system clock. System-wide synchronization mechanism is not always possible to implement in the distributed ad hoc network setting.

Bluetooth [3] is a new TDMA wireless technology that enables the formation of ad hoc networks called *scatternets*. Bluetooth has the interesting feature of not supporting a global slot synchronization mechanism. Instead, time slot reference is provided locally for each link by one of the node endpoints acting as master, while the other endpoint acts as slave. In this *asynchronous TDMA* setting, slots are wasted when nodes switch time references as slaves. This phenomenon has been reported in the scatternet scheduling literature [4, 5, 6, 7, 8] as a source of overhead. However, no formal study has examined its effect on the system's ability to allocate bandwidth. This ability is linked to the determination of the region of feasible allocations or, equivalently, to the solution of the related link schedule optimization problem.

*An earlier version of this paper has been presented at the *International Conference on Mobile Multimedia Communications (MoMuC) 2003*. This work has been supported by the Air Force Office of Scientific Research (AFOSR) under contract F49620-01-1-0197.

Due to the slots wasted for time reference alignment, the minimum period required for realizing a given allocation will be greater than the one required by a perfectly synchronized system. This increase can be seen as overhead due to system asynchronicity. Based on this observation, we can use a two-step procedure to address the optimal link scheduling problem for asynchronous TDMA ad hoc networks. The first step involves finding a minimum-period synchronized schedule for the demand allocation at hand. The second step, our contribution, utilizes the optimal synchronized schedule to find an asynchronous schedule of minimum overhead.

The amount of overhead depends on the order by which links are activated in the reference synchronized schedule. We first introduce an algorithm that derives a minimum-overhead asynchronous schedule for a specific ordering. The generated overhead is always upper-bounded regardless of ordering or network configuration. Using this algorithm, it is possible to determine the optimal solution by searching over all possible orderings. This leads to a combinatorial problem where exhaustive search is not feasible for large problem sizes. To this end, we introduce a heuristic algorithm of reduced complexity. The heuristic performs excellent for problem sizes where an optimal solution can be computed. When this is not possible, we investigate the effect of various system parameters on the generated overhead and use the upper bound as the performance measure.

The paper is organized as follows: section 2 introduces a conflict-free scheduling framework for asynchronous TDMA ad hoc networks. In sections 3, 4 and 5 the problem is introduced and formulated and the overhead minimization algorithms are presented. Section 6 evaluates the algorithms performances in various scenarios. section 7 places the proposed framework in the specific context of Bluetooth. Section 8 concludes the paper.

2 ASYNCHRONOUS TDMA COMMUNICATION MODEL

Every wireless node has a hardware clock that determines the timing of the radio transceiver. The clocks of different nodes are not synchronized and no mechanism exists for synchronizing them under a global time reference. The ad hoc network is represented as a directed graph $G(N, E)$. A directed edge from node i to node j signifies that i and j are within range and communicate on a link where i has been assigned the role of master and j the role of slave.

The system is slotted and carries point-to-point traffic—each transmission slot carries a packet destined to a single outgoing link. The time slot reference of each link is provided locally by the hardware clock of the master node endpoint. 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.

Each node has a single radio transceiver and can communicate (transmit or receive) to at most one link at a time. Thus, nodes need to coordinate their presence on links in mutual time intervals. Based on its own hardware clock, each node i divides time in fixed-size slots—each equal to the duration of a full-duplex communication slot. Transmissions on adjacent links are coordinated using a local link schedule \mathcal{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 (start acting as master or slave) or remain idle.

Local schedules of different nodes are not necessarily time-aligned. Every node i maintains a *relative phase* $\phi_{i \rightarrow j}$ with respect to each adjacent link (i, j) . If $\phi_{i \rightarrow j} = -1$, slot p in the local schedule \mathcal{S}_i overlaps in time with slots $(p - 1, p)$ in the local schedule \mathcal{S}_j . If $\phi_{i \rightarrow j} = 1$, then slot p in \mathcal{S}_i overlaps with slots $(p, p + 1)$ in \mathcal{S}_j . A relative phase $\phi_{i \rightarrow j} = 0$ indicates that the hardware clocks of the endpoints happen to be perfectly synchronized. The relative phase maintained at the other link endpoint j is $\phi_{j \rightarrow i} = -\phi_{i \rightarrow j}$. Given the relative phases and master-slave role assignment on link l , the *link phase* ϕ_l is defined as the relative phase of the master node endpoint.

Communication is successful on a link l only if both node endpoints assign time-overlapping slots in their local schedules. The assignment must be such that when the master starts polling in slot p of its local schedule, the slave must have assigned slots $p + \frac{\phi_l(1+\phi_l)}{2} - 1$ and $p + \frac{\phi_l(1+\phi_l)}{2}$ in its own local schedule for listening to this master. 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 aligning to the time reference of this master. In general, an extra slot is needed every time a node switches to a new time reference as slave.

We assume that the system does not support capture—if a node receives more than one transmissions at a specific time instant all of them are lost and a conflict occurs. The interference constraints determine which links can transmit simultaneously without causing conflicts at the intended receivers. According to *primary interference constraints*, links sharing common nodes cannot be activated simultaneously. This is due to the single radio transceiver (a node cannot transmit or receive simultaneously), as well as the point-to-point traffic requirement (each transmission is addressed to a single link). *Secondary interference* arises due to the broadcast nature of the wireless medium and occurs between different transmitter/receiver pairs. Two simultaneous transmissions, one from node i to node j and one from node k to node l will result to a secondary conflict if at least one of the receivers is within range of the sender of the other intended transmission.

The system may use a single channel or multiple channels for communication. A channel can be implemented as a distinct frequency or spread spectrum code. In single-

channel systems both primary and secondary interference exist. In multi-channel systems secondary interference can be mitigated by assigning to each link a channel derived by the (unique) identity of the master node endpoint. In this setting, every set of simultaneous transmissions satisfying the primary constraints will occur at different channels. Bluetooth is an instance of a multi-channel asynchronous TDMA system where communication channels (termed as *piconets*) are implemented as frequency hopping sequences.

When we refer to multi-channel systems, we will assume that different channels are orthogonal—transmissions on a channel are correctly received by a node listening on that channel despite any in-range transmissions that may be occurring at different channels. Recent studies [9] have indicated that this is a good approximation for the frequency hopping channels used by Bluetooth.

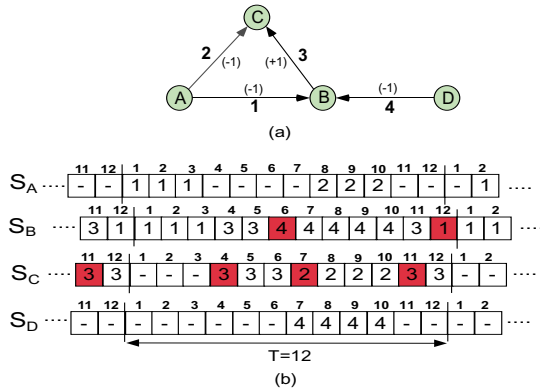


Figure 1: (a) Network configuration: Directed edges denote master-slave relationships. Nodes A and D act as masters on all their adjacent links, B is slave on links 1,4 and master on link 3 and C acts as slave on all its links. The numbers in parentheses denote link phases. As an example, since link 1 has a link phase of (-1), slot p in the local schedule S_A of master A must overlap with slots $(p - 1, p)$ in the local schedule S_C of slave C. (b) The asynchronous TDMA schedule refers to a multi-channel system: only links 2 and 4 can transmit simultaneously on the channels defined by nodes A and D, respectively. Slots where nodes switch time reference as slaves are marked in red. The realized slot allocation is $\tau = (\tau_1, \tau_2, \tau_3, \tau_4) = (3, 3, 3, 4)$.

For both single-channel and multi-channel systems, a slot allocation $\tau = [\tau_l]$ realized by the network TDMA schedule 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 schedule of the master endpoint.

A network configuration consists of the ensemble of a network topology, link phases and master-slave link role assignments. Figure 1 illustrates an example of a network configuration and asynchronous schedule as well as the slot allocation realized by this schedule.

3 OPTIMAL TDMA SCHEDULING

Given a network configuration it would be of interest to determine feasibility of any given demand slot allocation. This problem is equivalent to finding a link schedule of minimum period that realizes the demand allocation at hand. If the minimum period does not exceed the system period T_{system} the demand allocation is feasible. Although certain demand allocations can be realized by schedules of greater period than the minimum, the minimum period solution is optimal in the sense that it can detect *all* feasible allocations.

When global slot synchronization exists, the problem of finding the minimum-period TDMA link schedule for a demand allocation in an arbitrary topology is NP-complete, for both single channel [2] and multi-channel [10] systems. References [11, 12] propose efficient heuristics for this problem. Alternatively, optimal solutions exist for restricted topologies. For single channel systems, tree topologies can be optimally scheduled [13]. For multi-channel systems, scheduling links is equivalent to coloring edges in a multi-graph where the multiple edges between two node endpoints map to the slot requirement of the corresponding link. If the network topology is bipartite the optimal solution can be reached using minimum edge-coloring algorithms for bipartite multi-graphs [14].

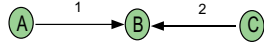
It is not straightforward to apply graph coloring techniques to the asynchronous TDMA setting. For example, in multi-channel systems there exists no one-to-one mapping of the slot demand per node pair in the network topology graph to multiple edges for this node pair in the corresponding multi-graph: in the asynchronous system, each link slot demand is the number of slots that should be allocated in the local schedule of the master endpoint; however the slave endpoint must allocate additional slots in its local schedule for time reference alignment. Also, as will be evident later in the discussion, the number of additional slots required in the slave local schedules depends on the order links are activated in the local schedules of the masters.

Our approach is based on the observation that the additional slots needed by the slaves yield an increase in period with respect to the minimum period of a perfectly synchronized system. This period increase is an overhead induced by the system asynchronicity. The link schedule optimization problem for the asynchronous system is translated to an overhead minimization problem: First, a synchronized link schedule that realizes the demand allocation is computed. Using this schedule as a reference we seek an asynchronous schedule of minimum overhead. If we can find a reference synchronized schedule of minimum period, a minimum overhead asynchronous schedule is a minimum-period asynchronous schedule. When the reference schedule period is sub-optimal, a minimum-overhead asynchronous schedule is still useful: the resulting period will be compared to T_{system} for determining feasibility of the demand allocation at hand. Therefore, minimum-

overhead schedules allow detection of a greater number of feasible allocations.

The amount of overhead depends on the ordering of link activations in the reference synchronized schedule. Consider the 3-node configuration of Figure 2 where node B is slave to both nodes A and C and where the demand allocation is 3 slots for each link.

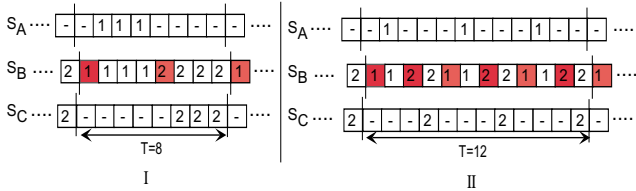
First, let us assume existence of slot synchronization. Since each node can communicate to only a single link at a time, the demand allocation can be realized by a minimum-period schedule of 6 slots. In this schedule, each link is activated 3 times by assigning concurrent slots in the endpoints' local schedules. Figure 2(b) illustrates two possible instances of the minimum-period schedule, each using a different ordering of link activations.



(a) Network configuration: B is a slave to both A and C .



(b) Synchronized system: Two possible synchronized schedule instances realizing slot allocation $(\tau_1, \tau_2) = (3, 3)$ in a minimum period of 6 slots



(c) Asynchronous system: Depending on the order of link activations, slot allocation $(\tau_1, \tau_2) = (3, 3)$ is realized by schedules of different minimum periods.

Figure 2: An example of the asynchronicity overhead

Figures 2(c)-I and 2(c)-II are two asynchronous schedules where links are activated in the order of Figures 2(b)-I and 2(b)-II, respectively. Both asynchronous schedules need a greater period than 6 slots to realize the demand allocation: in Figure 2(c)-I node B switches time reference only once per link yielding a period of 8 slots; in Figure 2(c)-II node B is forced to switch time reference every slot, yielding a period of 12 slots.

In the example of Figure 2, it is possible to determine

by inspection the link ordering and asynchronous schedule that yield minimum overhead (Schedule 2(c)-I). However, for arbitrary configurations and demand allocations a systematic approach is needed. We first introduce an algorithm that finds a minimum overhead asynchronous schedule for a fixed ordering of link activations in the reference synchronized schedule. This algorithm can be used to determine the minimum-overhead schedule over all possible orderings via exhaustive search. The following sections describe in detail our approach for the solution of this problem.

4 EQUIVALENT SCHEDULES

A *link activation set* consists of links that can simultaneously transmit without conflicts to the intended receivers. A *synchronized link schedule* \tilde{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{S} :

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

where π is a mapping of the indices $\{1, \dots, \tilde{T}\} \rightarrow \{1, \dots, \tilde{T}\}$.

Let $\tilde{S}^{(\pi)}$ be a synchronized schedule instance realizing allocation τ . For the ordering of link activations in $\tilde{S}^{(\pi)}$, allocation τ can be realized by more than one asynchronous schedules, each having a different period.

Consider the synchronized schedule instance of figure 2(b)-I that realizes allocation $(\tau_1, \tau_2) = (3, 3)$ by activating each link in 3 consecutive slots. For this ordering of link activations, the asynchronous schedule of figure 2(c)-I realizes the same allocation using a period of 8 slots. If slave B spent 5 slots instead of 4 listening on link 1, the same demand allocation would also be realized with this ordering of link activations but the overall period of the resulting asynchronous schedule would be 9 slots instead of 8.

We define an asynchronous schedule $S^{(\pi)}$ to be *equivalent* to a synchronized schedule instance $\tilde{S}^{(\pi)}$ if the following conditions hold:

- **(E.1):** Every node activates its adjacent links in $S^{(\pi)}$ in the same order as in $\tilde{S}^{(\pi)}$.
- **(E.2):** $S^{(\pi)}$ realizes the same allocation as $\tilde{S}^{(\pi)}$.
- **(E.3):** $S^{(\pi)}$ satisfies (E.1) and (E.2) in minimum period.

Thus, an equivalent schedule $S^{(\pi)}$ of a synchronized schedule instance $\tilde{S}^{(\pi)}$ is an asynchronous schedule that yields minimum overhead for the ordering of link activations in $\tilde{S}^{(\pi)}$.

We now present an algorithm called EQUIVALENT that takes as input a network configuration and a reference synchronized schedule instance $\tilde{\mathbf{S}}^{(\pi)}$ and outputs the equivalent asynchronous schedule $\mathbf{S}^{(\pi)}$ of $\tilde{\mathbf{S}}^{(\pi)}$.

EQUIVALENT constructs $\mathbf{S}^{(\pi)}$ incrementally by iterating over the link activation sets of $\tilde{\mathbf{S}}^{(\pi)}$. 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 $\mathbf{S}_i^{(\pi)}$ and $\mathbf{S}_j^{(\pi)}$, respectively ($p_n^{(0)} = 0, \forall n \in N$).

First, master i determines the earliest possible slot $p_i^{(k)}$ to be assigned to link l in $\mathbf{S}_i^{(\pi)}$. There are three possible cases:

- **Case A: Link l was activated in iteration $k - 1$:** The local schedules are "in synch" and node i can allocate to link l the next slot:

$$p_i^{(k)} = p_i^{(k-1)} + 1 \quad (2)$$

- **Case B: Link l was not activated in iteration $k - 1$ and $p_i^{(k-1)} > p_j^{(k-1)}$:** The master's local schedule is considered forward in time with respect to the slave's. The earliest slot is again:

$$p_i^{(k)} = p_i^{(k-1)} + 1 \quad (3)$$

- **Case C: Link l was not activated in iteration $k - 1$ and $p_j^{(k-1)} \geq p_i^{(k-1)}$:** The slave's local schedule is considered forward in time with respect to the master, so the master must find the earliest unassigned slot in $\mathbf{S}_i^{(\pi)}$ whose start time exceeds the end time of slot $p_j^{(k-1)}$ in $\mathbf{S}_j^{(\pi)}$:

$$p_i^{(k)} = p_j^{(k-1)} + \frac{\phi_l^2 - \phi_l + 2}{2}, \phi_l \in \{1, 0, -1\} \quad (4)$$

Then i assigns slot $p_i^{(k)}$ to link l . If any intermediate unassigned slots exist between $p_i^{(k-1)}$ and $p_i^{(k)}$, they are assigned as idle in $\mathbf{S}_i^{(\pi)}$.

Once the master updates its local schedule, slave j determines $p_j^{(k)}$ as the earliest unassigned slot in $\mathbf{S}_j^{(\pi)}$ whose end time exceeds the end time of $p_i^{(k)}$ in $\mathbf{S}_i^{(\pi)}$. Depending on the link phase ϕ_l the position of this slot is computed as:

$$p_j^{(k)} = p_i^{(k)} + \frac{\phi_l(1 + \phi_l)}{2}, \phi_l \in \{1, 0, -1\} \quad (5)$$

If there are any unassigned slots between $p_j^{(k-1)}$ and $p_j^{(k)}$, they are assigned to link l in $\mathbf{S}_j^{(\pi)}$.

The same assignment steps are performed for every link l in $A_{\pi(k)}$. For every node n not considered during iteration

k , $p_n^{(k)} = p_n^{(k-1)}$. At the end of iteration k , the *forward progress* $f(k)$ is defined as:

$$f(k) = \max_{n \in N} \{p_n^{(k)}\} \quad (6)$$

After \tilde{T} iterations, the asynchronous schedule period $T^{(\pi)}$ is set to the forward progress $f(\tilde{T})$. Then, starting again from $A_{\pi(1)}$, a few extra iterations are performed 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. An example of the algorithm operation is illustrated in Figure 3.

Due to the periodic nature of the reference TDMA schedule, EQUIVALENT requires at most $2\tilde{T}$ iterations. The maximum size for each link activation set is $N/2$ —the size of a perfect matching. Hence, addition of each link activation set requires $O(N)$ basic operations. Therefore, the computational complexity of EQUIVALENT is $O(N\tilde{T})$ (Due to space restrictions, we refer the reader to technical report [17] for the detailed proof).

For any network configuration and any link activation ordering π , EQUIVALENT possesses two important properties, summarized by the following theorems (established in [17]):

Theorem 1: The asynchronous schedule $\mathbf{S}^{(\pi)}$ derived by EQUIVALENT incurs minimum overhead for the link activation ordering corresponding to $\tilde{\mathbf{S}}^{(\pi)}$.

Theorem 2: If \tilde{T} is the period of the reference synchronized schedule, the period $T^{(\pi)}$ of any equivalent asynchronous schedule is upper bounded by $2\tilde{T}$.

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

Corollary on feasibility: Consider an asynchronous TDMA ad hoc network operating with a period T_{system} and a demand allocation τ . If τ can be realized by a *synchronized* schedule of period $\tilde{T} \leq \lfloor T_{system}/2 \rfloor$, then τ is feasible by the asynchronous system.

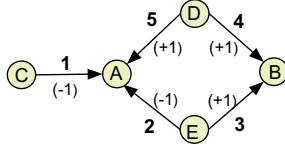
The corollary (also established in [17]) asserts that EQUIVALENT can realize at least half the allocations that are feasible under perfect synchronization. If the condition $\tilde{T} \leq \lfloor T_{system}/2 \rfloor$ holds for a demand allocation, any reference synchronized schedule instance can be used to generate an asynchronous schedule realizing this allocation. Otherwise, we must solve the optimization problem addressed next.

5 COMPUTING OPTIMAL ASYNCHRONOUS SCHEDULES

5.1 OPTIMAL ALGORITHM

The optimal asynchronous schedule can be determined by executing EQUIVALENT for all $\tilde{T}!$ synchronized schedule instances $\tilde{\mathcal{S}}^{(\pi)}$ and selecting the equivalent schedule of minimum overhead. Such an 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 consider only reference schedules where all instances of each link activation set are scheduled in consecutive slots—no switching slots are generated by EQUIVALENT when $A_{\pi(k-1)} = A_{\pi(k)}$; the overhead is zero during such a transition. If $M(\tilde{\mathcal{S}})$ is the set of *distinct* link activation sets appearing in the reference schedule, we only need to search $|M(\tilde{\mathcal{S}})|!$ schedule instances instead of $\tilde{T}!$. Unfortunately, even $|M(\tilde{\mathcal{S}})|$ can be prohibitively large for exhaustive searches. In this case we resort to the heuristic algorithm introduced in the next section.



(a) Network configuration

	9	10	1	2	3	4	5	6	7	8	9	10	1	2
A	2	1	1	5	5	5	1	1	1	1	2	1	1	5
B	4	4	3	3	3	3	4	4	3	4	4	4	3	3
C	-	1	1	-	-	-	1	1	1	1	-	1	1	-
D	4	4	-	5	5	5	4	4	-	4	4	4	-	5
E	2	-	3	3	3	3	-	-	3	-	2	-	3	3

T=10

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

S _A	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	1	5	5	5	5	1	1	1	1	1	2	2	1	1	1	5	5
S _B	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	3	3	3	3	3	4	4	4	3	3	4	4	4	4	3	3	3
S _C	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	1	-	-	-	-	1	1	1	1	-	-	-	1	1	-	-	-
S _D	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
	-	5	5	5	-	4	4	-	-	-	4	4	4	-	-	5	
S _E	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
	3	3	3	3	-	-	-	3	-	-	2	-	-	3	3	-	

(c) Numbers in parentheses indicate iteration where the slot was assigned on each node's local schedule. Switching slots are shaded. The equivalent schedule period (=14) is determined at the 10th iteration. Two additional iterations are performed so that all nodes assign 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 $p_n^{(k)}$ and $f(k)$.

5.2 MIN_PROGRESS

MIN_PROGRESS is a heuristic for overhead minimization that consists of two phases. Phase I determines an ordering π_h of the distinct link activation sets in $M(\tilde{\mathcal{S}})$. Phase II involves two steps: first, a synchronized schedule instance is formed, where distinct link activation sets are ordered according to π_h and the instances of each set are activated in consecutive slots. Second, this synchronized schedule instance is input to EQUIVALENT to generate the final asynchronous schedule.

We now describe Phase I that selects 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 link activation set of $M(\tilde{\mathcal{S}})$ 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 ($U^{(0)} = M(\tilde{\mathcal{S}})$). 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 and set M^{α_k} is removed from the U -set. The same steps are repeated until the U -set becomes empty after $|M(\tilde{\mathcal{S}})|$ iterations.

Phase I can be extended to select and insert multiple link activation sets per iteration, according to a horizon parameter h . During iteration k , all possible h -set blocks in the U -set and all possible orderings ($h!$) of the link activation sets within each h -set block are considered. The block and ordering that yields minimum forward progress

Figure 3: An example of the EQUIVALENT algorithm execution

is selected and added to the asynchronous schedule. The selected block is removed from the U-set and the next iteration is performed. Depending on whether h divides $|M(\tilde{\mathcal{S}})|$ or not, the algorithm will terminate in $\lfloor \frac{|M(\tilde{\mathcal{S}})|}{h} \rfloor$ or $\lfloor \frac{|M(\tilde{\mathcal{S}})|}{h} \rfloor + 1$ iterations, respectively.

Proposition: For $h \geq 1$ and fixed, the computational complexity of MIN_PROGRESS is $O(N|M(\tilde{\mathcal{S}})|^{h+1})$.

Proof: The complexity of MIN_PROGRESS is determined by the complexities of Phases I and II. Let $M = |M(\tilde{\mathcal{S}})|$. During iteration k of Phase I, $\binom{M-(k-1)h}{h}$ blocks are considered and, for each block, $h!$ orderings of activation sets are tested. Testing each ordering involves insertion of h activation sets to the asynchronous schedule. Therefore, the total number of insertions C_I throughout the execution of Phase I will be:

$$C_I = \sum_{k=1}^{\lfloor \frac{M}{h} \rfloor} \binom{M-(k-1)h}{h} \cdot h! \cdot h + r(M, h) \cdot h \quad (7)$$

where

$$r(M, h) = \begin{cases} (M \bmod h)! & \text{if } M \bmod h \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

After algebraic manipulations, equation (7) becomes:

$$C_I = h \sum_{k=0}^{\lfloor \frac{M}{h} \rfloor - 1} \prod_{i=0}^{h-1} ((M-i) - hk) + r(M, h) \cdot h \quad (9)$$

An upper bound to C_I is:

$$\begin{aligned} C_I &< h \sum_{k=0}^{\lfloor \frac{M}{h} \rfloor - 1} \prod_{i=0}^{h-1} (M - hk) + r(M, h) \cdot h \\ &= h \sum_{k=0}^{\lfloor \frac{M}{h} \rfloor - 1} (M - hk)^h + r(M, h) \cdot h \\ &= h \sum_{k=0}^{\lfloor \frac{M}{h} \rfloor - 1} \sum_{i=0}^h (-1)^i a_i M^{h-i} (hk)^i + r(M, h) \cdot h \\ &= h \sum_{i=0}^h (-1)^i a_i h^i M^{h-i} \sum_{k=0}^{\lfloor \frac{M}{h} \rfloor - 1} k^i + r(M, h) \cdot h \\ &= h \sum_{i=0}^h (-1)^i a_i h^i M^{h-i} \sum_{k=1}^{\lfloor \frac{M}{h} \rfloor - 1} k^i + r(M, h) \cdot h \end{aligned} \quad (10)$$

where a_i are positive integers. The term $r(M, h) \cdot h$ is constant, since $(M \bmod h) < h$. The term $\sum_{k=1}^{\lfloor \frac{M}{h} \rfloor - 1} k^i$ is $\Theta(M^{i+1})$ because the power sum $\sum_{k=1}^n k^i$ can be expanded

in polynomial form as $\sum_{j=0}^{i+1} b_j n^j$, where b_j are integers.

Multiplying this term with M^{h-i} in (10), causes the entire term in (10) to be $\Theta(M^{h+1})$.

A lower bound to C_I is:

$$C_I > h \sum_{k=0}^{\lfloor \frac{M}{h} \rfloor - 1} \prod_{i=0}^{h-1} ((M - (h-1)) - hk) + r(M, h) \cdot h \quad (11)$$

and can be shown to be $\Theta(M^{h+1})$ in a similar way as (10). Therefore, C_I is $\Theta(M^{h+1})$.

Since insertion of an activation set requires $O(N)$ operations, the complexity of Phase I is $O(NM^{h+1})$. The complexity of Phase II is $O(NM)$ because it uses EQUIVALENT on the ordering π_h computed by Phase I. Hence, overall complexity of MIN_PROGRESS is dominated by Phase I and is $O(NM^{h+1})$.

Given a specific reference schedule $\tilde{\mathcal{S}}$, the horizon h may range from 1 to $|M(\tilde{\mathcal{S}})|$. Larger values of h are expected to yield better performance because more orderings are tested per iteration. This comes at the expense of increased computational complexity. For the maximum horizon value, MIN_PROGRESS is essentially the optimal algorithm—it includes a single iteration where a block of $|M(\tilde{\mathcal{S}})|$ orderings must be exhaustively tested. Thus, h must be carefully selected to ensure tractability. According to MIN_PROGRESS, the maximum number of $\binom{|M(\tilde{\mathcal{S}})|}{h}$ blocks must be considered in the first iteration. The horizon h must be small enough to allow exhaustive enumeration of this number, as well as exhaustive enumeration of $h!$ orderings per block. The effect of h in performance will be investigated in the experiments section that follows.

6 PERFORMANCE EVALUATION

6.1 FACTORS AFFECTING THE OVERHEAD

We are interested in evaluating performance in view of the factors that affect the asynchronicity overhead. The overhead is first related to the topology structure. In general, denser topologies are expected to produce higher overhead because more links will translate to a higher number of time reference switches. Performance is also affected by the master-slave role assignments. In the example of Figure 2, if node B is assigned as master to nodes A and C , the overhead is zero due to the single time reference in the system.

For a specific network configuration the overhead also depends on the demand allocation at hand. A parameter specific to the demand allocation is the ratio $|M(\tilde{\mathcal{S}})|$ of distinct link activation sets to the period \tilde{T} of the optimal reference schedule. A small ratio is desirable because overhead

is generated only during the transitions between distinct activation sets in the synchronized schedule. Another related parameter is the period \tilde{T} of the synchronized schedule. Larger periods may allow for smaller $|M(\tilde{S})|/\tilde{T}$ ratios and, therefore, less generated overhead.

6.2 EXPERIMENTAL SETTING

Performance must be evaluated for a variety of network configurations and optimal reference synchronized schedules. As mentioned in section 3, determination of optimal synchronized schedules is in general an NP-complete problem. However, for bipartite topologies in multi-channel systems, the minimum period equals the maximum node utilization:

$$\tilde{T}(\tau) = \max_{i \in N} \sum_{l \in L(i)} \tau_l. \tag{12}$$

where $L(i)$ is the set of adjacent links to node i . Thus, in this case, optimal reference synchronized schedules of period \tilde{T} can be constructed by generating arbitrary conflict-free schedules where at least one node transmits during the entire period.

In the experiments we consider $|N|$ -node multi-channel bipartite networks 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 channel degree parameter B_{max} is an upper bound on the number of channels a node can participate as slave. Such a constraint would arise in practice to avoid excessive overhead. We also restrict the number of links where a node can act as master to 7. This restriction is specific to Bluetooth, a multi-channel asynchronous TDMA system. 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 $f \times 100\%$ links of the baseline topology remain intact while the rest have been removed.

Given a topology constructed as above, asynchronicity is introduced by 1) master-slave role assignments on the links and 2) arbitrary phase differences on the hardware clocks of the nodes in the network. According to the link role assignments, a node may act as master to all its adjacent links (termed as "master") or act as slave to all its adjacent links (termed as "S/S bridge"), or act as master to some links and slave to others (termed as "M/S bridge").

6.3 PERFORMANCE OF MIN_PROGRESS WITH RESPECT TO OPTIMAL

Six 20-node bipartite topologies (10 masters and 10 S/S bridges) of varying density 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 4 compares the resulting optimal and MIN_PROGRESS periods. For each topology, the periods are averaged over all reference schedules. Using a horizon $h = 1$, MIN_PROGRESS exceeds the optimal by less than one slot on the average, while in topology 5 it exceeds the optimal by 1.3 slots on the average.

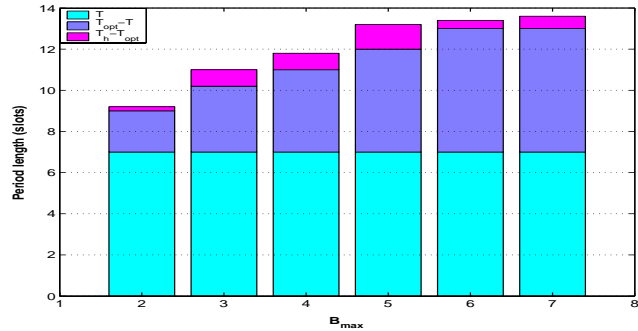


Figure 4: MIN_PROGRESS vs. optimal. Each bar graph corresponds to a different 20-node bipartite network configuration where density increases by varying B_{max} from 2 to 7. The reference synchronized schedule period is 7 slots. The optimal period T_{opt} and the MIN_PROGRESS ($h = 1$) period T_h of each bar are averages of 100 reference synchronized schedules.

The optimal and MIN_PROGRESS periods increase with B_{max} and for $B_{max} = 7$ they both approach 14 slots, the upper bound of EQUIVALENT. The high overhead stems from B_{max} being equal to the small reference period \tilde{T} : S/S bridges with such a channel degree need to switch time reference almost every slot regardless the link activation order in the reference schedule.

6.4 PERFORMANCE OF MIN_PROGRESS FOR LARGE PROBLEM SIZES

For each parameter set $(N, B_{max}, f, \tilde{T})$ we generate 10 topologies and, for each topology, 100 arbitrary reference synchronized schedules. For each $(N, B_{max}, f, \tilde{T})$, the overhead is averaged over the corresponding topologies and reference schedules and is plotted as the % increase in the reference period \tilde{T} . If T_h is the period computed by MIN_PROGRESS, this quantity is equal to $\frac{T_h - \tilde{T}}{\tilde{T}}$, with 100% denoting that MIN_PROGRESS yields period $2\tilde{T}$, the upper bound of EQUIVALENT.

We proceed by investigating the various factors that affect the performance of MIN_PROGRESS.

6.4.1 Effect of horizon

In this set of experiments, we use 20-node bipartite topologies (10 masters and 10 S/S bridges) and vary the density parameter f ($B_{max} = 7$) and reference period \tilde{T} . Figure 5 plots the overhead of MIN_PROGRESS using up to 3 activation sets per block.

For all scenarios, the overhead decreases as h increases. The improvement is always more drastic from $h = 1$ to $h = 2$ than from $h = 2$ to $h = 3$. Using $h = 2$ instead of $h = 1$ appears beneficial for larger periods and densities (bar graphs $f = 0.6, 0.9$ in figure 5(c) and figure 5(c)), with a maximum overhead reduction of 13% at $\tilde{T} = 112$ and $f = 0.9$. For the lowest density considered, MIN_PROGRESS performs similarly for all h . Overall, a horizon $h = 2$ seems to provide a good performance/complexity trade-off at higher reference periods and topology densities, while a horizon $h = 1$ appears sufficient at low topology densities.

6.4.2 Effect of phase and role assignments

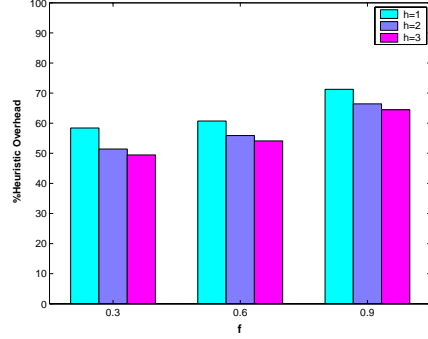
Consider a topology graph $G(N,E)$. Since for every link l , ϕ_l can be $-1, 0$, or 1 , there are $3^{|E|}$ possible link phase assignments in the network. Also, there are $2^{|E|}$ possible master-slave link role assignments.

In this experiment, we consider 20-node bipartite topologies. For a specific topology and reference synchronized schedule, we measure the standard deviation of the generated overhead of MIN_PROGRESS for a sample of 1000 arbitrary phase (or role) assignments. Then, for each parameter set (f, \tilde{T}) we plot the average standard deviation over the corresponding topologies and reference schedules.

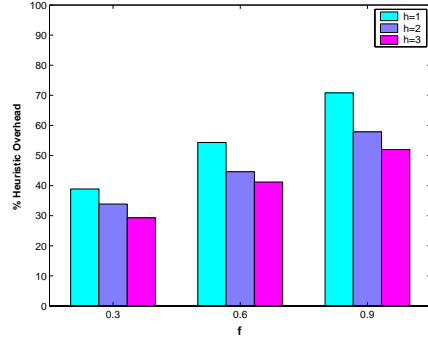
For every (f, \tilde{T}) , role variability (Fig 7) produces higher standard deviation than phase variability (Fig 6)—the difference never exceeds 1%. Apart from this difference, both figures have similar properties: For a fixed density the standard deviation appears insensitive to \tilde{T} —less than 0.5% changes are observed. However, for every \tilde{T} , the standard deviation decreases as the density increases. This indicates that the overhead deviates less from a certain mean as the number of links per locality increases; therefore variability in phase and role assignments affect the algorithm performance to a lesser extent in this case.

6.4.3 Effect of density

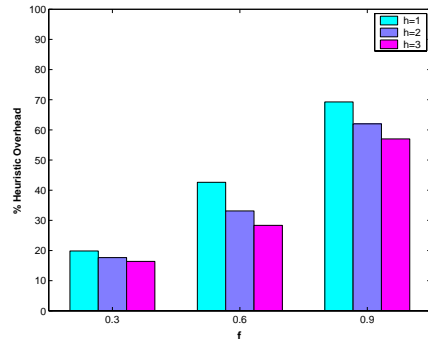
Here, a 100-node (50 masters, 50 S/S bridges) baseline bipartite topology is used. Figure 8 illustrates the effect of B_{max} on the overhead of MIN_PROGRESS. For fixed \tilde{T} the overhead consistently increases with B_{max} . At $\tilde{T} = 28$, the overhead is 15% when $B_{max} = 2$ but reaches 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 indicates that a non-negligible overhead may still exist even if the system uses a large period. Similar trends arise in Figure 9 where $B_{max} = 7$ and only parameter f is used to vary the topology density. The overhead increases with network density regardless of the number of



(a) $\tilde{T} = 28$ slots, $B_{max} = 7$, f varies.



(b) $\tilde{T} = 112$ slots, $B_{max} = 7$, f varies.



(c) $\tilde{T} = 448$ slots, $B_{max} = 7$, f varies.

Figure 5: Effect of the choice of horizon for varying topology densities and reference periods ($N = 20$, $B_{max} = 7$).

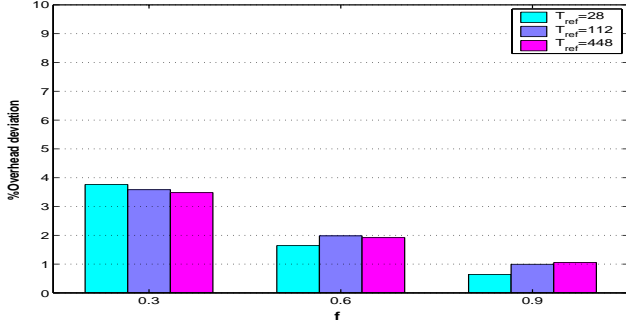


Figure 6: Average overhead standard deviation due to link phase variability for 20-node networks and various values of f and \tilde{T} ($h = 1$, fixed link roles per topology)

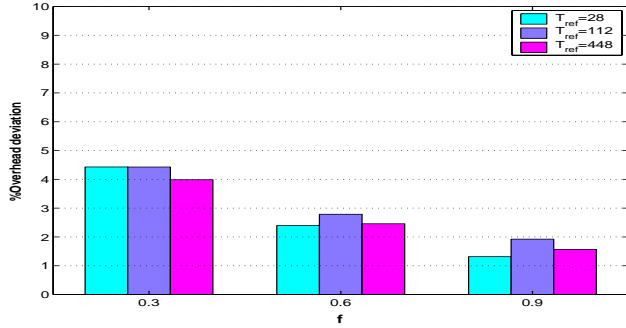


Figure 7: Average overhead standard deviation due to link role assignment variability for 20-node networks and various values of f and \tilde{T} ($h = 1$, fixed link phases per topology).

time references in which each node participates.

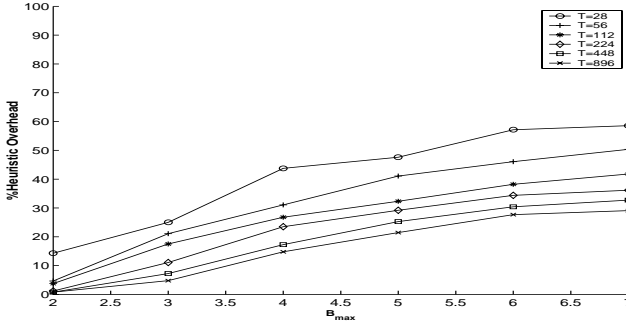


Figure 8: Overhead of MIN_PROGRESS ($h = 1$) for 100-node networks as B_{max} and \tilde{T} vary ($f = 1.0$)

6.4.4 Effect of demand allocation

The previous experiments investigated the algorithm performance averaged over arbitrary demand allocations and topologies. A natural question that arises next is whether there exists a network configuration and demand allocation for which the generated overhead is maximized. In this section we make a first attempt to informally classify such worst-case instances and then test our intuition through simulations.

Let the topology be bipartite and $\Psi(\tilde{T})$ be the set of all allocations realized by a synchronized schedule of mini-

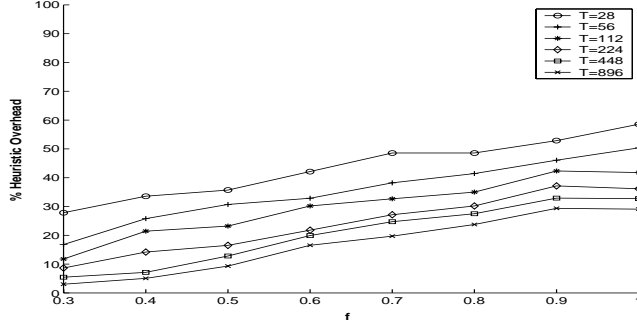


Figure 9: Overhead of MIN_PROGRESS ($h = 1$) for 100-node networks as f and \tilde{T} vary ($B_{max} = 7$).

period \tilde{T} . For any allocation τ in $\Psi(\tilde{T})$, let $BN(\tau)$ be the set of nodes that receive maximum utilization \tilde{T} under τ .

$$BN(\tau) = \{n : arg \max_{i \in N} \sum_{j \in N(i)} \tau_{ij}\}. \quad (13)$$

We conjecture that maximum overhead will be generated if the following conditions hold for a demand allocation τ^{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 has maximum link degree.
- **P2:** The node is a S/S bridge.
- **P3:** Allocation τ^{max} is such that the node is requested to allocate an equal number of slots to its adjacent links.

A maximum utilization node will be considered at every iteration of an overhead minimization algorithm. Also, since this is a node of maximum degree and acts as a S/S bridge, it will visit the maximum possible number of time references (B_{max}) as slave. If link demands are equal for this node, we can show that the overhead will be maximized under the worst ordering of link activations.

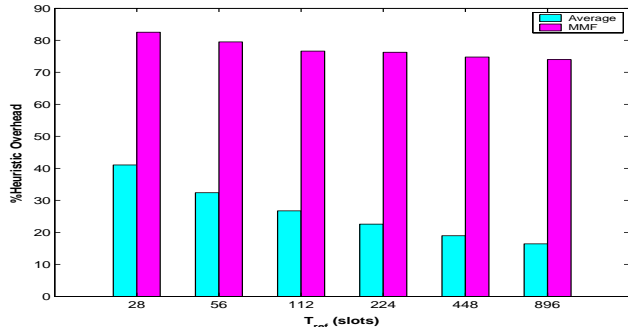


Figure 10: MIN_PROGRESS overhead for maxmin fair allocations vs. average MIN_PROGRESS overhead. For each reference period, both quantities are averaged over all topologies considered in Figures 8 and 9.

A maxmin fair allocation in a synchronized multi-channel wireless ad hoc network maximizes utilization of

the nodes with maximum link degree [15, 16]. If at least one of these nodes is also assigned as a S/S bridge then conditions P1-P3 will hold.

Figure 10 compares the MIN_PROGRESS overhead resulting from a maxmin fair reference schedule and the average MIN_PROGRESS overhead over 100 arbitrary schedules. (The algorithm in [15] is used to compute the reference maxmin fair schedules).

The average MIN_PROGRESS overhead decreases as the system period increases. The overhead for the maxmin fair schedule however, does not change significantly—in the order of 80% for all cases. This indicates that the overhead can be very high for the allocations we identified even if we use an overhead minimization algorithm such as MIN_PROGRESS. Counter-intuitively, the overhead remains high even if the reference period increases. Nevertheless, it is always less than the upper bound given by EQUIVALENT.

7 THE CASE OF BLUETOOTH

Bluetooth is a new wireless technology that enables the formation of ad hoc networks called scatternets. A scatternet is an instance of a multi-channel asynchronous TDMA ad hoc network where channels are implemented as frequency hopping sequences (termed as piconets), and each node can act as master to at most seven adjacent links.

The scatternet scheduling problem is the problem of coordinating the node visits in different piconets in an efficient manner and is currently a subject of intense research effort. Emphasis is placed on distributed schemes—the approaches can be categorized according to the degree of coordination they offer. “Hard” coordination schemes [8, 15] provide deterministic allocations via conflict-free scheduling; however, a certain degree of implementation complexity and communication overhead is needed to maintain the conflict-free property under topology dynamics. “Soft” coordination schemes [5, 6, 7] trade off perfectly conflict-free transmissions for lower complexity. The downside is loss of the ability to provide bandwidth allocation guarantees.

While there is still a simplicity v.s. performance debate between the two approaches, the overhead incurred when nodes switch time references always exists. In this paper we introduced a hard-coordination framework for overhead minimization for two reasons. First, in this case, the overhead is naturally linked to the ability of the system to allocate bandwidth. Second, a useful point of reference is established since conflict-free scheduling is the best we can do to minimize overhead. The derivation of a similar overhead minimization framework for soft coordination schemes would be an interesting research avenue to pursue.

8 CONCLUSIONS

In this paper, we addressed for the first time the problem of minimizing overhead in TDMA wireless ad hoc networks that use multiple local time slot references instead of a single global time slot reference. This overhead arises due to slots wasted when nodes synchronize to the different local time slot references and manifests as loss of supported allocations with respect to a perfectly synchronized system. The problem was cast and addressed using a generic framework and the results can be directly applied for the case of Bluetooth, a wireless technology operating according to the asynchronous TDMA communication paradigm.

It was demonstrated that the overhead can significantly affect the ability of a network to allocate bandwidth if no measures are taken to minimize it. We introduced two scheduling algorithms that aim to minimize overhead while ensuring that the generated overhead has an upper bound regardless of the network configuration 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 such cases an efficient heuristic was devised. We also identified and verified, through simulations, certain conditions on demand allocations and network configurations for which the overhead can be high even if an overhead minimization algorithm is run. Further investigation of the exact nature of such conditions is an interesting research direction.

Both optimal and heuristic algorithms are centralized and can operate in settings where global information is available. More importantly, they can be used to provide design insights and serve as a reference performance measure for any overhead-aware distributed approaches that will follow.

Manuscript received on . . .

REFERENCES

- [1] B. Hajek and G. Sasaki, Link scheduling in polynomial time. In *Proc. IEEE Transactions on Information Theory*, Vol. 34, pg. 910–917, September 1988.
- [2] E. Arıkan, Some complexity results about packet radio networks. In *Proc. IEEE Transactions on Information Theory*, Vol. 30, pg. 681–685, July 1984.
- [3] Bluetooth Special Interest Group, Specification of the Bluetooth system, version 1.0. In *www.bluetooth.com*, 2003.
- [4] G. Miklos, Z. Turanyi, A. Valko and P. Johansson, Performance Aspects of Bluetooth Scatternet Formation. In *Proc. ACM MOBIHOC 2000*, August 2000, Boston, MA, USA.
- [5] N. Johansson, F. Alriksson, and U. Jonsson, JUMP mode: a dynamic window-based scheduling framework for Bluetooth scatternets. In *Proc. ACM MOBIHOC 2001*, October 2001, Long Beach, CA, USA.

- [6] A. Racz, G. Miklos, F. Kubinszky and A. Valko, A Pseudo Random Coordinated Scheduling algorithm for Bluetooth Scatternets In *Proc. ACM MOBIHOC 2001*, October 2001, Long Beach, CA, USA.
- [7] S. Baatz, M. Frank, C. Kuhl, P. Martini and C. Scholz, Bluetooth Scatternets: An Enhanced Adaptive Scheduling Scheme In *Proc. IEEE INFOCOM*, June 2002, New York, NY, USA.
- [8] N. Johansson, U. Korner and L. Tassiulas, A distributed scheduling algorithm for a Bluetooth scatternet. In *Proc. International Teletraffic Congress (ITC)*, 2001, Salvador da Bahia, Brazil.
- [9] A. Kumar and R. Gupta, Capacity Evaluation of Frequency Hopping Based Ad-hoc Systems In *Proc. ACM SIGMETRICS*, June 2001, Cambridge, MA, USA.
- [10] I. Holyer, The NP-completeness of edge coloring In *Proc. SIAM Journal of Computing*, Vol. 10, pg. 169–197, 1981.
- [11] M. Post, P. Sarachik and A. Kershenbaum, A Biased Greedy Algorithm for Scheduling Multihop Radio Networks In *Proc. Annual Conference on Information Sciences and Systems (CISS)*, March 1985, Johns Hopkins University, Baltimore, MD, USA.
- [12] S. Ramanathan, A Unified Framework and Algorithm for Channel Assignment in Wireless Networks In *Proc. IEEE INFOCOM*, September 1997, Kobe, Japan.
- [13] S. Ramanathan and E. Lloyd, Scheduling algorithms for multihop radio networks In *Proc. IEEE/ACM Transactions on Networking*, Vol. 1, pg. 166–177, April 1993.
- [14] N. Alon, A simple algorithm for edge-coloring bipartite multigraphs In *Proc. Information Processing Letters*, Vol. 85, pg. 301–302, September 2003.
- [15] T. Salonidis and L. Tassiulas, Distributed On-Line Schedule Adaptation for balanced slot allocation in Bluetooth Scatternets and Other Ad Hoc Network Architectures. In *Technical Report TR 2002-24, Institute of Systems Research (ISR)*, 2002, University of Maryland, College Park.
- [16] L. Tassiulas and S. Sarkar, Maxmin Fair Scheduling in Wireless Networks In *Proc. IEEE INFOCOM*, June 2002, New York, NY, USA.
- [17] T. Salonidis and L. Tassiulas, Asynchronous TDMA: Scheduling and Performance In *Technical Report TR 2002-52, Institute of Systems Research (ISR)*, 2004, University of Maryland, College Park.