A Distributed Scheduling Algorithm for Wireless Networks with Constant Overhead and Arbitrary Binary Interference.

Jean-Claude Bermond[1], Dorian Mazauric[1,2], Vishal Misra[3], Philippe Nain[2]

1 : Mascotte, INRIA, I3S(CNRS/Univ of Nice-Sophia), Sophia Antipolis, France
2 : Maestro, INRIA, Sophia Antipolis, France
3 : Columbia University, New York, USA

This work investigates distributed transmission scheduling in wireless networks. Due to interference constraints, "neighboring links" cannot be simultaneously activated, otherwise transmissions will fail. Hereafter, we consider any binary model of interference. We follow the model described by Bui, Sanghavi, and Srikant in [3]. We assume. There are two phases during each slot: first the control phase which determines what links will be activated next, followed by a transmission phase during which data is transmitted. We assume random arrivals of data (packets) on each link during each slot, so that a buffer (queue) is associated to each link. It takes exactly one time slot to transmit a packet on a link. Since nodes do not have a global knowledge of the network, our aim (like in [3]) is to design for the control phase a distributed algorithm which identifies a set of non-interfering links.

For example, in the primary node interference model (where two links interfere only if they are incident), a set of non interfering edges is called a *matching*. In order to ensure the largest stability region of the system we want to choose links so that the sum of their weight (queue-length) is as large as possible so as to realize a *maximum matching*. The idea behind the maximum matching is to decrease the largest queues.

Centralized algorithms have been proposed to solve this problem both for random arrivals in [7] and deterministic arrivals in [6]. To be efficient the control phase should be as short as possible; this is done by exchanging control messages during a constant number of mini-slots (constant overhead).

In this work we design the first fully distributed local algorithm with the following properties: it works for any arbitrary binary interference model; it has a constant overhead (independent of the size of the network and of the queue-lengths) and it requires no knowledge. Indeed the algorithm in [5] does not have a constant overhead, whereas the one described in [3] is only valid for the primary node interference model. Furthermore, both algorithm need to know the queue lengths of the "neighboring links", which are difficult to obtain in a wireless network with interference. We prove that our algorithm gives a maximal set of active links (i.e. in each interference set, there is at least one active edge). We also give sufficient conditions for stability under Markovian assumptions. Finally the performance of our algorithm (throughput, stability) is investigated and compared via simulations to that of previously proposed schemes.

Let us sketch the main ideas of our algorithm `Algorithm Log`. Details, proofs and examples can be found in the full version [1].

We use a binary symmetric interference model and define the interference set of a link $e \in E$, denoted by $\varepsilon(e)$, as the set of edges interfering with e. We denote by $c(e)$, the capacity of the edge e, that is the number of packets that e can serve during one time slot if the link e is active. Let $q_t(e)$ (also called the weight of link e) be the number of packets in the queue of link $e \in E$ at the beginning of step $t \geq 1$. `Algorithm Log` concerns only the control phase which determines at each step the set of active links (on which data will be sent during the data phase). The control phase is divided into mini-slots.

A first idea consists in associating virtual weights (see below) to current queue-lengths. These virtual weights will be used to determine in each mini-slot which edges will send a control message. Only edges with weight greater than their capacity ($\frac{q_t(e)}{c(e)} \geq 1$) will participate in `Algorithm Log`. We then split the values of $\frac{q_t(e)}{c(e)}$ into a small number K of disjoint intervals (classes) $I_0, \ldots, I_{K-1}$ with the constraint that the largest class I_{K-1} contains all the values greater than some value L: $I_{K-1} =]L, \infty[$. In the example (see Figure 1), we have taken $c(e) = 1$, $L = 100$ and $K = 5$; the five classes are associated to the values of $q_t(e)$ as follows: $I_0 = [1, 10]$; $I_1 =]10, 30]$; $I_2 =]30, 60]$; $I_3 =]60, 100]$ and $I_4 =]100, \infty[$.

When K is small, many interfering edges might belong to the same interval. To differentiate them, we also give to each edge a color (integer), $\gamma(e)$, in such a way that two interfering edges have different colors. Let C be the number of colors. In order to get different rankings in consecutive slots, we will permute the colors. To this end, we associate in the slot t the value $\gamma_t(e) = (\gamma(e) + t - 2)_{mod C} + 1$. Figure 1(b) shows a coloring for a cycle of length 9, with $C = 3$, when the primary node interference model is used (i.e. two edges interfere if they are incident). The coloring is given at some step t; the first edge will get color 3 at step $t + 1$, then color 1 at step $t + 2$, color 2 again at step $t + 3$ and so on.

Finally, we assign to an edge of class i a virtual weight $q'_t(e) = C * i + \gamma_t(e)$. The virtual weights satisfy the two properties: for any edge e, $0 < q'_t(e) \leq CK \quad \forall t \geq 1$, and if $e' \in \varepsilon(e)$ (and $e \in \varepsilon(e')$) then, $q'_t(e) \neq q'_t(e')$. Let $T = \lceil log_2(CK) + 1 \rceil$. We associate to $q'_t(e)$ a *control vector* $v_{t,e} = (v_{t,e}(1), \ldots, v_{t,e}(T))$ where $v_{t,e}(i)$, $1 \leq i \leq T$, corresponds to the ith bit of $q'_t(e)$. Table 1 gives these values for the example in Figure 1(c).

Another main idea of `Algorithm Log` is to use interference as information. More precisely at each mini-slot of `Algorithm Log`, an edge can be active, inactive, or still undetermined. The control phase is itself divided into α subphases. The first subphase consists of T mini-slots. During a mini-slot i, $1 \leq i \leq T$, $e \in E$ sends a control message if and only if e is still undetermined and $v_{t,e}(i) = 1$. We have the three rules:

(a) if e sends a control message and e does not receive one from an

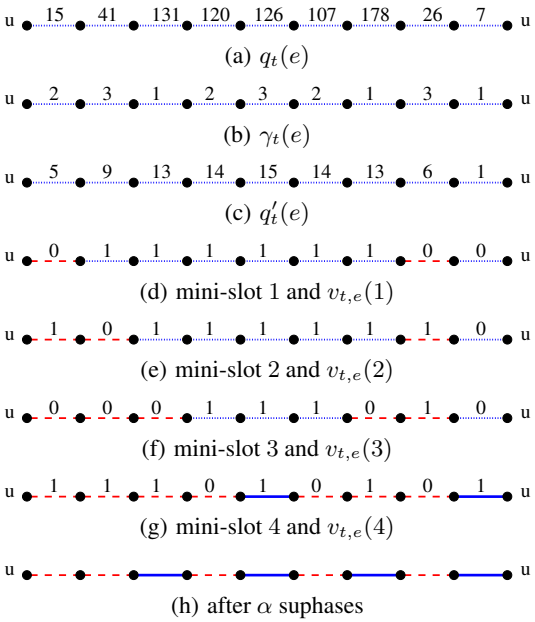

Figure 1: `Algorithm Log` for a cycle: C=3, K=5, L=100

edge in $\varepsilon(e)$, then e becomes **active**;
(b) if e receives a control message (coming from an edge in $\varepsilon(e)$) and does not send a control message, then e becomes **inactive**;
(c) otherwise e remains **undetermined**.

At the end of subphase 1 (after T mini-slots), we get a valid set of active links, meeting the interference constraints. Figure 1(d-g) shows the application of `Algorithm Log` on the example. One can prove furthermore that there is no undetermined edge (see [1]). After subphase 1, it may happen that the set of active links is not maximal (edges can be added to this set without removing current edges in this one). This is the case in the example in Figure 1(g), where the two active edges do not form a maximal matching. To deal with this situation, we apply $\alpha - 1$ times the scheme described for subphase 1, but only with edges which can be added, that is inactive edges without active links in their interference sets (Figure 1(h)). We prove that choosing $\alpha = T$, `Algorithm Log` always computes a maximal set of active edges.

THEOREM 1. *If we choose $\alpha = T = \lceil log_2(CK) + 1 \rceil$ in `Algorithm Log`, then for any edge e, with positive virtual weight, there exists one edge $e' \in \varepsilon(e) \cup \{e\}$ such that e' is active at the end of `Algorithm Log`.*

From this theorem, we can deduce that the number of mini-slots of the control phase of `Algorithm Log` is $T_{log} = \lceil log_2(CK) + 1 \rceil^2 + \lceil log_2(CK) + 1 \rceil - 1$. Therefore, we have interest to minimize K and C. The determination of C is related to coloring problems; in particular, in the case of the primary node interference model, minimizing C corresponds to computing the Edge Chromatic Number (see [4]). If we fix in advance the number T_{log} of mini-slots allowed, we can compute the maximum possible value of K. We can also play with the value of L.

In [1], we analyze the stability of `Algorithm Log`. Under the assumptions that the arrival processes are independent renewal processes, we establish a sufficient stability condition by using the approach in [2, Theorem 1]. To state this condition, let $A_t(e)$ be the number of arrivals on link e in slot t. Define $H(e)$ as the maximum number of links that can be scheduled if link e is not scheduled and let $H := \max(1, \max_{e \in E} H(e))$. We show in [1] that the load vector $\mathbf{b} = (E[A_t(e)], e \in E)$ stabilizes `Algorithm Log` if there exists $\epsilon > 0$ such the vector $CH/(C + H - 1))(\mathbf{b} + \epsilon \mathbf{c})$ belongs to the capacity region introduced by Tassiulas et al [8], where $\mathbf{c} := (c(E), e \in E)$ is the link capacity vector. We conjecture that if for any edge e, $\sum_{e' \in \varepsilon(e) \cup \{e\}} E[A_t(e')] < \min_{e' \in \varepsilon(e) \cup \{e\}} c(e')$ then `Algorithm Log` is stable.

				Control Vector: $v = v_{t,e}$			
$q_t(e)$	class	$\gamma_t(e)$	$q'_t(e)$	$v(1)$	$v(2)$	$v(3)$	$v(4)$
$[1, 10]$	I_0	1	1	0	0	0	1
$[1, 10]$	I_0	2	2	0	0	1	0
$[1, 10]$	I_0	3	3	0	0	1	1
$]10, 30]$	I_1	1	4	0	1	0	0
$]10, 30]$	I_1	2	5	0	1	0	1
$]10, 30]$	I_1	3	6	0	1	1	0
$]30, 60]$	I_2	1	7	0	1	1	1
$]30, 60]$	I_2	2	8	1	0	0	0
$]30, 60]$	I_2	3	9	1	0	0	1
$]60, 100]$	I_3	1	10	1	0	1	0
$]60, 100]$	I_3	2	11	1	0	1	1
$]60, 100]$	I_3	3	12	1	1	0	0
$]100, \infty[$	I_4	1	13	1	1	0	1
$]100, \infty[$	I_4	2	14	1	1	1	0
$]100, \infty[$	I_4	3	15	1	1	1	1

Table 1: $(q'_t(e), v_{t,e})$ for every possible pair $(q_t(e), \gamma_t(e))$ for `Algorithm Log`: C=3, K=5, L=100

In [1] we report many simulation results which confirm this conjecture and show a very good behavior of the algorithm and better than that of [3]. In particular `Algorithm Log` gives a set of active edges of large weight and the size of the queues remains small.

Acknowledgment: this work has been partially supported by région PACA, ANR AGAPE and DIMAGREEN, and European project IST FET AEOLUS.

1. REFERENCES

[1] J.-C. Bermond, D. Mazauric, V. Misra, and P. Nain. Distributed call scheduling in wireless networks. Technical Report RR-6763, INRIA, Dec. 2008.

[2] V. Bhandari and N. H. Vaidya. A result on hybrid scheduling in wireless networks. Technical report, University of Illinois, Dept. Electrical and Computer Eng., March 2009.

[3] L. X. Bui, S. Sanghavi, and R. Srikant. Distributed link scheduling with constant overhead. *IEEE/ACM Transactions on Networking*, 17(5):1467–1480, 2009.

[4] S. Fiorini and R. J. Wilson. *Edge-colourings of graphs*. Pitman, 1977.

[5] A. Gupta, X. Lin, and R. Srikant. Low-complexity distributed scheduling algorithms for wireless networks. In *INFOCOM*, pages 1631–1639, 2007.

[6] R. Klasing, N. Morales, and S. Pérennes. On the complexity of bandwidth allocation in radio networks. *Theoretical Computer Science*, 406(3):225 – 239, 2008.

[7] L. Tassiulas. Scheduling and performance limits of networks with constantly changing topology. *IEEE Transactions on Information Theory*, 43(3):1067–1073, 1997.

[8] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Conference on Decision and Control*, pages 2130–2132 vol.4, 1990.