Some Game-Theoretic Problems in Wireless Ad-Hoc Networks

E. Altman[1], Vivek S. Borkar[2], Arzad A. Kherani[1], P. Michiardi[3], and R. Molva[3]

[1] INRIA, 06902 Sophia Antipolis, France
{Eitan.Altman,alam}@sophia.inria.fr
[2] School of Technology and Computer Science, Tata Institute of Fundamental Research, Mumbai, 400 005, India
borkar@tifr.res.in
[3] GET/EURECOM, Sophia-Antipolis, France
{Pietro.Michiardi,molva}@eurecom.fr

Abstract. Wireless Ad-hoc networks are expected to be made up of energy aware entities (nodes) interested in their own perceived performance. We consider a simple random access model for a wireless ad hoc network to address problems of finding an optimal channel access rate and providing incentive for cooperation to forward other nodes' traffic. By casting these problems as noncooperative games, we derive conditions for the Nash equilibrium and provide distributed algorithms to learn the Nash equilibrium.

Keywords. Game theory, Stochastic approximation algorithm.

1 Introduction

Wireless ad hoc networks (also referred to as packet radio networks and multihop radio networks) consist of mobile nodes communicating over a shared wireless channel. Contrary to cellular networks, where the nodes are restricted to communicate with a set of carefully placed base stations, in wireless ad hoc networks there are no base stations; any two nodes are allowed to communicate directly if they are close enough. A wireless ad hoc network can be considered as a system of various mobile wireless devices that is dynamically changing and self-organizing in arbitrary networks. These wireless devices are usually constantly changing their location because they are, for example, carried by people (for example, PDAs). These devices need to form a dynamic network without a pre-existing communication infrastructure. For such networks we address two problems mentioned in the subsections below.

1.1 Optimal Channel Access Rate

These networks are expected to use medium access protocols similar to the IEEE 802.11 protocol [12]. The 802.11 protocol is inherently a random access protocol

G. Kotsis and O. Spaniol (Eds.): Mobile and Wireless Systems, LNCS 3427, pp. 82–104, 2005.

since the protocol running in each wireless device keeps a backoff timer whose mean value essentially denotes the node's willingness to attempt a transmission, thus also risking a collision. Clearly, the nodes can not have a very large mean backoff timer value as this can add significantly to the delay in their packet transmission and also the node may miss opportunities of transmission. On the other hand, if all the nodes set their mean backoff timer value at a very small value, there will be singnificant amount of collision, thus again having an adverse effect on the nodes' performance. It is thus clear that there is some optimal attempt probability for the wireless devices at which the risk of collision could be balanced by the benefit of successfull transmission (and hence less delay) while making use of the most of the transmission opportunities available. Since, in such networks the nodes are rational, i.e., a node wants to maximize its own performance, a node needs to compute its own attempt probability such that it gets best performance for what other nodes do.

Much of the work on wireless ad hoc networks (with some exceptions) has been on the protocol design issues for medium access (the various variants of the IEEE 802.11 protocol, for example [5]) or dynamic routing in such networks [13,19]. The authors of [8] also look at the problem of tuning of the IEEE 802.11 parameters but they assume that all the nodes are cooperative and do not consider the rational behaviour of the nodes. Relatively little has been done on the resolution of 'optimal' self-organization as an optimization problem which amounts, roughly, to finding optimum parameters for the various protocols. As an example of the latter, consider [14] where the problem of adapting transmission attempt probabilities is viewed as a single optimization problem with a stated performance metric to be minimized and the optimization task is executed by a stochastic gradient method implemented in a distributed fashion. This approach, though suitable for sensor networks where each wireless device (or sensor) cooperates to acheive a common goal, is not suitable for a wireless ad hoc network with rational entities.

In Section 3 and 4, we view the problem as a noncooperative game with each node trying to optimize its own objective. By assuming a particularly simple performance metric for each node we get an explicit characterization of a Nash equilibrium. This in turn can be adaptively learnt by an iterative scheme that uses local information exchange. The required information for the algorithm should be available from a standard topology learning procedure.

1.2 Incentive for Forwarding

In order to maintain connectivity in an Ad-hoc network, mobile terminals should not only spend their resources (battery power) to send their own packets, but also for forwarding packets of other mobiles. Since Ad-hoc networks do not have a centralized base-station that coordinates between them, an important question that has been addressed is to know whether we may indeed expect mobiles to collaborate in such forwarding. If mobiles behave selfishly, they might not be interested in spending their precious transmission power in forwarding of other mobile's traffic. A natural framework to study this problem is noncooperative

game theory. As already observed in many papers that consider noncooperative behavior in Ad-hoc networks, if we restrict to simplistic policies in which each mobile determines a fixed probability of forwarding a packet, then this gives rise to the most "aggressive" equilibrium in which no one forwards packets, see e.g. [11, Corollary 1], [18], thus preventing the system to behave as a connected network. The phenomenon of aggressive equilibrium that severely affects performance has also been reported in other noncooperative problems in networking, see e.g. [10] for a flow control context (in which the aggressive equilibrium corresponds to all users sending at their maximum rate).

To avoid very aggressive equilibria, in Section 5 we propose strategies based on threats of punishments for misbehaving aggressive mobiles, which is in the spirit of a well established design approach for promoting cooperation in Ad-hoc networks, carried on in many previous works [11,24]. In all these references, the well known "TIT-FOR-TAT" (TFT) strategy was proposed. This is a strategy in which when a misbehaving node is detected then the reaction of other mobiles is to stop completely forwarding packets during some time; it thus prescribes a threat for very "aggressive" punishment, resulting in an enforcement of a fully cooperative equilibrium in which all mobiles forward all packets they receive (see e.g. [11, Corollary 2]). The authors of [22] also propose use of a variant of TFT in a similar context.

2 A Model for Wireless Ad Hoc Networks

We consider a wireless adhoc network where all the terminals transmit on a common carrier frequency so that a terminal can either receive or transmit at a time. The transmission range of a terminal is denoted by R_0. Henceforth we use the terms terminal and node to mean the same entity.

We assume that the condition for node j to successfully decode the transmission of its neighbouring node i is that none of the other neighbours of node j transmit when i is transmitting.

A node is assumed to be in exactly one of the following modes at any time:

1. be transmitting to its neighbours, or,
2. sensing the channel for the transmission from its neighbour, or,
3. can be in the *sleep mode*, i.e., it is neither transmitting nor receiving and also not wasting its battery power in sensing the channel. Note that in this mode of operation a node may lose an opportunity of reception of a transmission from one of its neighbours.

Time is assumed to be slotted and channel access is random, i.e., in each slot, a node i either decides to transmit with probability α_i, or enters sleep mode with probability γ_i or decides to receive with probability $(1-\gamma_i-\alpha_i)$. α_i is called the *attempt probability* of node i, $\alpha_i + \gamma_i \leq 1$.

A node can be *heard* by only certain nodes called the neighbours. The neighbours of a node i are the nodes within a distance R_0 from node i. A node is assumed to have some amount of data in its transmit queue at all times. For ease

of presentation, we first assume that a transmission of a data packet from node i is considered successful iff *all* the neighbours of node i can decode the transmission successfully. Later we remove this restriction and show how to accommodate the possibility that a node i transmits to a specific neighbouring node j with probability $\alpha_{i,j}$. In the latter case a transmission is declared successful if the node that the transmission was destined for can decode the transmission successfully.

We assume that the topology of the network is fixed but the nodes need not be aware of the complete topology. The network consists of N nodes. Denote the set of nodes by $\mathcal{N}$. For $i,\ j \in \mathcal{N}$ say $i \rightarrow j$ if node i can receive node j's transmission, i.e., node j is within a distance of R_0 from node i. For ease of presentation we assume in this section that $i \rightarrow j$ iff $j \rightarrow i$, i.e., R_0 is same for all the nodes; we use the notation $i \leftrightarrow j$ to mean either of these two. Associate with each node $i \in \mathcal{N}$, a neighbourhood set $\mathcal{N}(i) := \{j \in \mathcal{N} : i \leftrightarrow j\}$. Denote the node incidence matrix thus obtained by Φ, i.e.,

$$\Phi(i,j) = \Phi(j,i) = \begin{cases} 1 \text{ if } \quad i \leftrightarrow j \quad \text{or} \quad j \in \mathcal{N}(i) \\ 0 \text{ otherwise.} \end{cases} \tag{1}$$

We assume that Φ is an irreducible matrix, i.e., every node $i \in \mathcal{N}$ has a *path* to any other node $j \in \mathcal{N}$; this amounts to assuming that the network is connected.

3 Finding Optimal Channel Access Rates

Assume for now that $\gamma_i = 0$ for all nodes i in the network, i.e., in any slot a node is either transmitting a packet (which contains an update information) or is listening to the channel. It is clear from the model described above that if α_i is very small for all the nodes, there will be significant delay in information processing though the battery power consumption will be less. On the other hand increasing α_is to large values will result in increased collisions in the network and will also waste the battery power in unsuccessful transmissions. Thus there is a need to find the optimum value of these attempt probabilities.

This phenomenon is very similar to that of finding the arrival rates in a slotted Aloha system for multiple access (see [4]). It is well known (see [4]) that for such a system there exists an optimal attempt rate at which the system throughput is maximized.

Our problem here is to find, for a given network, the values of α_i (or, $\alpha_{i,j}$, as the case may be) which maximize a performance measure to be defined in Section 3.1. The objective then is to come up with an algorithm using which a node i computes the optimal attempt probability (α_i or $\alpha_{i,j}$) for itself *in a distributed manner*.

3.1 Optimization Problem

For simplicity of presentation, in this section we assume that a transmission from a node is successful if all of its neighbours receive it correctly. In later sections we show how to modify the problem formulation and solution of this section for

the case where in any slot a node i transmits to a particular neighbouring node j with probability $\alpha_{i,j}$.

Let α_i, $i \in \mathcal{N}$, be the probability that node i transmits in any slot. A transmission from node i will be successful iff

1. None of the nodes belonging to the neighbourhood set of i transmit, and
2. For each $k \in \mathcal{N}(i)$ none of the nodes belonging to the neighbourhood set of k (except node i) transmit.

The second condition above means that none of the second hop neighbours of i transmit. By second hop neighbours we mean $\cup_{j\in\mathcal{N}(i)}\mathcal{N}(j)\backslash(i \cup \mathcal{N}(i)) =: \mathcal{S}(i)$, i.e., the set of neighbours of neighbours of i excluding i and $\mathcal{N}(i)$. Let ζ be the second hop node incidence matrix, i.e., $\zeta(i,j) = 1 = \zeta(j,i)$ iff $j \in \mathcal{S}(i)$ and $\zeta(i,j) = 0$ otherwise.

Denote by $P_s(i)$ the probability that a transmission from node i is successful. It follows from the conditions mentioned above that

$$P_s(i) = \Pi_{j\in\mathcal{N}(i)}(1-\alpha_j)\Pi_{k\in\mathcal{S}(i)}(1-\alpha_k) \tag{2}$$

Thus the probability that a node i made a successful transmission in any slot is $\alpha_i P_s(i)$. We want to maximise this probability for all the nodes i while simultaneously reducing the probability of collision among the transmissions from node i (to minimize the battery power wasted in collisions) and also the probability of missing out on a transmission opportunity. Thus the problem for node i can be written as follows:

$$\text{Minimize} \qquad -A\alpha_i P_s(i) + B\alpha_i(1-P_s(i)) + C(1-\alpha_i)P_s(i) \tag{3}$$

$$\text{such that } \alpha_i \geq \alpha_{min} > 0 \tag{4}$$

$$\text{and } \alpha_i \leq \alpha_{max} < 1. \tag{5}$$

Here:

- the first term in the cost (3) is the negative of the 'reward' for successful transmission, the latter being $A > 0$ times the probability of a successful transmission,
- the second is the penalty for an unsuccessful attempt, being $B > 0$ times the probability of a collision,
- and the third is the penalty for lost opportunities, being $C > 0$ times the probability of no transmission when one was possible.

The bounds on α_i imposed by the equations (4) and (5) are to ensure the connectivity of the network. This is because a node with attempt probability 0 or 1 will be effectively cutoff from the rest of the network and will also lead to disconnectivity between other node pairs.

Since the nodes are each trying to optimize their own objectives without any cooperation, this is a noncooperative game. The 'action space' for each node is the interval $[\alpha_{min}, \alpha_{max}]$ from which it chooses the transmission probability. This is compact convex. Also, each node's objective function (3) is separately

convex continuous in each argument. Thus a standard argument based on the Kakutani fixed point theorem ensures the existence of a Nash equilibrium, i.e., a choice $\underline{\alpha}^* = [\alpha_1^*, \cdots, \alpha_N^*]$ such that if all but the $i-$th node transmit with probabilities α_j^*'s, $j \neq i$, then it is optimal for $i-$th node also to use α_i^* [23]. Our aim will be to attain this Nash equilibrium. With this objective, we first seek the necessary conditions for the Nash equilibrium.

Since for fixed values of α_j, $j \neq i$, this is a single agent optimization problem faced by the $i-$th node, we consider the corresponding Kuhn-Tucker conditions. Let $\theta = \frac{B}{A+B+C}$ and $\eta = \ln \frac{B}{A+B+C}$. For any vector $\underline{\alpha}$ of attempt probabilities, let $A^*(\underline{\alpha}) = \{i : \alpha_{min} < \alpha_i < \alpha_{max}\}$. The (equivalent of) Kuhn-Tucker necessary conditions [23] for a vector $\underline{\alpha}$ to be a Nash equilibrium, i.e., a componentwise local minimum of corresponding cost functions when the other components are unperturbed, are

$$\theta - P_s(i) = 0, \qquad \forall\ i \in A^*(\underline{\alpha}) \tag{6}$$

$$\theta - P_s(i) \geq 0, \qquad \forall\ i : \alpha_i = \alpha_{min} \tag{7}$$

$$\theta - P_s(i) \leq 0, \qquad \forall\ i : \alpha_i = \alpha_{max} \tag{8}$$

where $\theta - P_s(i)$ is in the direction of the gradient at point $\underline{\alpha}$ for the cost function of node i.

Let α^* be the Nash equilibrium for the game problem and assume, for simplicity, that $A^*(\alpha^*) = \mathcal{N}$. The case where $A^*(\alpha^*) \neq \mathcal{N}$ will be studied in a later section. Let $\beta_j = \ln(1 - \alpha_j)$. After taking logarithms, Kuhn-Tucker necessary conditions of (6) can be rewritten as,

$$\sum_{j \in \mathcal{N}(i) \cup \mathcal{S}(i)} \beta_j = \eta, \qquad \forall i \tag{9}$$

This set of equations can be written in matrix form as

$$\underline{\eta} - (\Phi + \zeta)\beta = 0, \tag{10}$$

where β and $\underline{\eta}$ are column vectors of same size with the j^{th} entry being β_j and η respectively.

This suggests the iteration

$$\underline{\beta}(n+1) = \underline{\beta}(n) + a(n)(\underline{\eta} - (\Phi + \zeta)\underline{\beta}(n)),$$

where $\{a(n)\}$ are the usual stochastic approximation stepsize schedules, i.e., positive scalars satisfying

$$\sum_n a(n) = \infty, \ \sum_n a(n)^2 < \infty.$$

By the standard 'o.d.e. approach' to stochastic approximation, this tracks the asymptotic behaviour of the o.d.e. [16]

$$\dot{x}(t) = \underline{\eta} - (\Phi + \zeta)x(t).$$

This is a linear o.d.e. which would indeed converge to the solution of (10) if it were stable. Unfortunately, the stability cannot be a priori assumed. Thus we consider the iteration

$$\underline{\beta}(n+1) = \underline{\beta}(n) + a(n)(\Phi + \zeta)(\underline{\eta} - (\Phi + \zeta)\underline{\beta}(n)), \tag{11}$$

corresponding to the o.d.e.

$$\dot{x}(t) = (\Phi + \zeta)(\underline{\eta} - (\Phi + \zeta)x(t)). \tag{12}$$

This will be stable if $(\Phi + \zeta)$ is nonsingular, whence $(\Phi + \zeta)^2$ will be positive definite. The solution will correspond to the linear system

$$(\Phi + \zeta)\underline{\eta} - (\Phi + \zeta)^2\beta = 0, \tag{13}$$

which then has the same solution as (10). Thus node i will be solving the i^{th} row of (13). This will need further modification when either the nonsingularity of $(\Phi + \zeta)$ does not hold or when one of the constraints on some α_i is active so that either (7) or (8) is operative. The basic iteration described above then needs to be modified.

Remark: There is one further complication that needs to be underscored, viz., that our distributed implementation cannot ensure that all components are updated equally often. Thus the theory of [6] suggests that the limiting o.d.e. will be not (12), but

$$\dot{x}(t) = \Lambda(t)(\Phi + \zeta)(\underline{\eta} - (\Phi + \zeta)x(t)), \tag{14}$$

where $\Lambda(t)$ is a diagonal matrix for each t with nonnegative entries $\{\lambda_i(t)\}$ on the diagonal. These reflect the differing comparative frequencies of updating for different components. (See [6] for details.) We shall assume that the latter are strictly positive, which means that the components get updated comparably often, though not necessarily equally often. In our case, this change of o.d.e. does not alter the conclusions. To see this, first note that (12) is of the form

$$\dot{x}(t) = -\nabla F(x(t)),$$

for $F(x) \stackrel{defn}{=} ||\underline{\eta} - (\Phi + \zeta)x||^2$. Thus $F(\cdot)$ itself serves as a 'Liapunov function' for it, with

$$\frac{d}{dt}F(x(t)) = -||\nabla F(x(t))||^2 \leq 0,$$

the equality holding only on the solution set of (13). When we replace (12) by (14), one has instead

$$\frac{d}{dt}F(x(t)) = -||\sqrt{\Lambda(t))}\nabla F(x(t))||^2 \leq 0,$$

leading to identical conclusions, whence the original convergence claims continue to hold.

3.2 The Algorithm

The following algorithm implements the iterations suggested in Section 3.1.

Algorithm 1

1. *Set the slot number* $n = 0$.
2. *Initialize* $\alpha_i^{(0)}$, $1 \leq i \leq N$, *to some small positive values. Also let* $N(i) = 0$, *the last slot number when node* i *updated its attempt probability.*
3. *For* $1 \leq i \leq N$, *node* i *does the following sequence of operations:*
   (a) *It either decides to transmit with probability* $\alpha_i^{(n)}$, *or decides to go into sleep mode with probability* γ_i, *or, if not any of the above, senses the channel for any transmission.*
   (b) *If decided to transmit, a node does the following:*
      i. *Sends the data packet (measurements) destined for all the neighbouring nodes.*
      ii. *It also transmits the information relevant for its nodes for updating their attempt probabilities based on (11). In particular, node* i *transmits* $\alpha_i^{(n)}$, $N(i)$, $\mathcal{N}(i)$ *and all the information (attempt probabilities and the last time the node updated its attempt probability) it has about the nodes it can reach in three hops.*
   (c) *If decided to sense the channel, a node does the following:*
      i. *If node* i *receives a signal that can be decoded correctly, then it checks if the received signal contains the update information. If yes, update node* i*'s local information about its four hop neighbours based on the information it extracted from the transmission received from its first hop neighbour. Here node* i *updates its estimate of* α_j *only if the value of* $N(j)$ *that it has now received is more than node* i*'s copy of* $N(j)$.
      ii. *Update* $\alpha_i^{(n)}$ *based on the updated information.*
      iii. *Set* $N(i) = n$.
4. $n = n + 1$.
5. *Go to step* 3.

Remarks:

1. This algorithm is similar to the DSDV protocol [19] for route discovery in the ad hoc 802.11 networks as each node keeps its own copy of the local information of the connectivity between the neighbouring 4 hop nodes and transmits a part of this information to the neighbouring nodes. This implies that mobility or failure of the nodes can also be taken into account in this algorithm as a node i can consider another node j (which is reachable in at most 4 hops from node i) as failed or nonexistent if node i's information about node j says that $N(j)$ has not been updated for a long time, say $M\lceil\frac{1}{\hat{\alpha}_j}\rceil$, where M is a large integer and $\hat{\alpha}_j$ is node i's most recent information about α_j.

2. The property of the nodes being dense is not required here and the exact physical distance between two nodes is no longer relevant now as the required information, i.e., $(\Phi + \zeta)^2$ is already obtained.
3. A node keeps information only about its four hop neighbours. The amount of storage required for this information grows with the density of the network (which should be true for any such algorithm). Note that the storage required for the required information does *not* change by increasing the span of the network for a fixed node density (i.e., increasing the area of the network by adding more nodes so as to keep the node density fixed). Compare this with DSDV or the algorithm of [14] where a node keeps information about the *complete* network.

3.3 Problem Formulation to Incorporate Sleep Mode

Algorithm 1 was for the case where, in any slot, a node either decides to transmit or else decides to receive. To accommodate for the possibility that a node i can decide to be in sleep mode with a fixed probability γ_i (known to node i), for this case equation (2) for $P_s(i)$ can be modified to

$$P_s(i) = \Pi_{j\in\mathcal{N}(i)}(1-\gamma_j-\alpha_j)\Pi_{k\in\mathcal{S}(i)}(1-\gamma_k-\alpha_k) \tag{15}$$

Similarly, in the penalty term for missed opportunities in equation (3), $(1-\alpha_i)$ should be replaced by $(1-\gamma_i-\alpha_i)$. Algorithm 1 needs to be modified to take care of this possibility. Now, a node i transmits γ_i along with its other update information meant for its neighbours. The rest of the algorithm works similarly.

One can also consider $\{\gamma_i\}$ as additional decision variables that can be tuned to find the optimal trade-off between sleep and alert modes. To do this, one may add to node i's 'cost' (3) the additional terms $D(1-\gamma_i) + G\gamma_i P_s(i),\ D, G > 0$. The first is the cost on battery power utilization (this could be fine tuned further to allow for different costs for different kinds of usage), the second is the cost of missed opportunities due to sleep mode. The algorithm can be easily modified to incorporate optimization over $\{\gamma_i\}$.

3.4 Problem Formulation for Transmissions Destined for Fixed Nodes

Algorithm 1 was for the case where a node's transmission is meant for all of its neighbouring nodes and a node i has only one attempt probability α_i. Now we show how to modify Algorithm 1 for the case where a node's transmission is destined for a particular node $j \in \mathcal{N}(i)$. In this case a node i, in any slot and *for each of its neighbour* $j \in \mathcal{N}(i)$, decides to send a packet destined for node j with probability $\alpha_{i,j}$ *independent of anything else*. Note that it is possible that node i decides to send data to two or more of its neighbours simultaneously in which case the transmission is unsuccessful. We again assume here that $\gamma_i = 0$. For this case (2) can be written as

$$\begin{aligned} P_s(i,j) = {} & \Pi_{l\in\mathcal{N}(i)\setminus j}(1-\alpha_{i,l})\Pi_{l\in\mathcal{N}(j)}(1-\alpha_{j,l})\Pi_{l\in\cup_{k\in\mathcal{N}(j)\setminus i}\mathcal{N}(k)\setminus i} \\ & \Pi_{m\in\mathcal{N}(j)\setminus i}(1-\alpha_{l,m}). \end{aligned} \tag{16}$$

Thus a node computes $\alpha_{i,j}$ using Algorithm 1 based on the information that it receives from its neighbouring nodes. Note that now a node sends the $\alpha_{i,j}$'s of its 3 hop neighbours instead of just the α_i's.

Numerical results based on an implementation of the algorithm are presented in [7].

4 Optimal Channel Access Rate with Reward on Reception

In the previous section we assumed that a node is interested in its performance as a traffic source. Typically a node in an ad hoc network is both sender as well as receiver of packets and hence will be interested in a *combined* performance measure that reflects its performance as a sender and as a receiver. These are two conflicting requirements: a node, if it tries to be too aggressive in sending packets, may lose opportunity to receive packets meant for itself and vice versa.

We now assume that nodes never go into sleep mode, i.e., they are either transmitting or ready to receive. The channel access is again random, i.e., in each slot, a node i decides to transmit (broadcast) with probability α_i and decides to receive with probability $(1-\alpha_i)$. The quantity α_i is called the *attempt probability* of node i. What follows can be easily modified to account for a node i keeping a attempt probability $\alpha_{i,j}$ for its neighboring node j, or to some subset of its neighbors (multicast).

Our problem now is to find, for a given network, the values of α_i (or $\alpha_{i,j}$, as the case may be) which maximizes node i's performance.

Let $P_s(i)$ be the conditional probability that a transmission attempt from node i is *successful* (conditioned on the event that node i transmits), and $P_r(j,i)$ denote the probability that a transmission from node j is successfully received by node i.

Here one can have various notions of node i's transmission being *successful*. We use the simple (though not restrictive) criteria: node i's transmission is successful if *all* of it's neighboring nodes correctly receive the transmission.

Each node wants to maximise its own utility function which reflects the performance obtained by the node under the sending probabilities selected by the nodes in the network. A common ingredient of the utility function of node i is a combination of the rates at which node i successfully transmits and receives packets. Thus the problem for node i is to maximise

$$U_i(\underline{\alpha}) = A_i\alpha_i P_s(i) + \sum_{j\in\mathcal{N}(i)} A_{i,j}\alpha_j P_r(j,i) - C_i\bar{\alpha}_i P_s(i) \tag{17}$$

such that $\alpha_i \geq 0$ and $\bar{\alpha}_i = 1 - \alpha_i \geq 0$. Here A_i, C_i and $A_{i,j}$ are some non-negative constants. The first and second terms here are "rewards" for *success* owing to, respectively, transmission and reception. $A_{i,j}$ will be zero for node j whose transmission can not be directly received by node i. The third term is included to act as a punishment for missed opportunities, thus aiming at

maximising node i's use of network. Note that the last term also is the probability of the event where none of the neighboring nodes of node i are sending to node i when node i is ready to receive, thus this term also represents the time wasted by node i in trying to receive when there is nothing to receive.

Our definition of $P_s(i)$ means that none of the first or second hop neighbors of i transmit when node i does. Thus

$$P_s(i) = \Pi_{j\in\mathcal{N}(i)}(1-\alpha_j)\Pi_{k\in\mathcal{S}(i)}(1-\alpha_k). \tag{18}$$

Similarly, it is seen that, for $j \in \mathcal{N}(i)$, $P_r(j,i)$ is,

$$P_r(j,i) = \Pi_{k\in\mathcal{N}(i)\cup\{i\}\setminus\{j\}}(1-\alpha_k). \tag{19}$$

This is again viewed as a concave N-person game thus a Nash equilibrium exists, i.e., a choice $\underline{\alpha}^* = [\alpha_1^*, \cdots, \alpha_N^*]$ such that if all but the $i-$th node transmit with probabilities α_j^*'s, $j \neq i$, then it is optimal for $i-$th node also to use α_i^* [20].

Again, since for fixed α_j, $j \neq i$, this is a single agent optimization problem faced by the $i-$th node, we consider the corresponding Kuhn-Tucker condition. For any vector $\underline{\alpha}$ of attempt probabilities, let $A^*(\underline{\alpha}) = \{i : 0 < \alpha_i < 1\}$. The (equivalent of) Kuhn-Tucker conditions for a vector $\underline{\alpha}$ to be a Nash equilibrium, i.e., a componentwise local maximum of corresponding utility functions when the other components are unperturbed, are (with $A_{i,i} := A_i + C_i$)

$$A_{i,i}P_s(i) - \sum_{j\in\mathcal{N}(i)} A_{i,j}\alpha_j \frac{P_r(j,i)}{1-\alpha_i} = 0, \qquad \forall\ i \in A^*(\underline{\alpha}) \tag{20}$$

$$A_{i,i}P_s(i) - \sum_{j\in\mathcal{N}(i)} A_{i,j}\alpha_j \frac{P_r(j,i)}{1-\alpha_i} \geq 0, \qquad \forall\ i : \alpha_i = 1 \tag{21}$$

$$A_{i,i}P_s(i) - \sum_{j\in\mathcal{N}(i)} A_{i,j}\alpha_j \frac{P_r(j,i)}{1-\alpha_i} \leq 0, \qquad \forall\ i : \alpha_i = 0. \tag{22}$$

Let α^* be a Nash equilibrium for the game problem and assume, for simplicity, that $A^*(\alpha^*) = \mathcal{N}$. Let $\underline{\alpha}$ be a column vector whose i^{th} entry is α_i. Also introduce $\mathbf{G}(\underline{\alpha}) := \frac{\partial}{\partial\alpha_i}U_i(\underline{\alpha}) = A_{i,i}P_s(i) - \sum_{j\in\mathcal{N}(i)} A_{i,j}\alpha_j\frac{P_r(j,i)}{1-\alpha_i}$.

4.1 Effect of Imposing Power Constraints

Till now we have not imposed any restriction on the possible values that α_i's are allowed to take (except that $\alpha_i \in [0,1]$). Since the nodes are battery power constrained, one would like to see the effect of imposing a constraint on α_i so as to use the battery power efficiently. A natural candidate for such a constraint for node i is $T_i\alpha_i + R_i(1-\alpha_i)P_r(i) \leq P_i$, where T_i and R_i are the average power required for transmission and reception of packets, P_i is the average battery power of node i and $P_r(i)$ is the probability that node i is trying to receive while it is not transmitting. $P_r(i) = 1 - \Pi_{j\in\mathcal{N}(i)}(1-\alpha_j)$, i.e., that a node spends R_i

amount of power whenever there is a transmission attempt from at least one neighboring node. In practice, the case of interest would be $T_i \geq P_i \geq R_i$. (If $P_i \geq \max(T_i, R_i)$ then, effectively, the α_i's are not battery power constrained.) Note now that the action space of the nodes are dependent on the actions of other nodes. The existence of a Nash equilibrium would follow if the constraint set so obtained is convex [20]. For a general network topology, it can be shown that the constraint defining functions $P_i - T_i\alpha_i - R_i(1-\alpha_i)P_r(i)$ are quasi-concave [17] so that the constraint set is convex. Further, the constraint set is easily seen to be nonempty because the point $\alpha_i = 0,\ \forall i$ is always feasible. Details of proof showing quasi-concavity of the power constraint functions is omitted.

4.2 A Distributed Algorithm

To compute α_i, the Kuhn-Tucker condition of Equation 20 suggests the following (gradient ascent type) iteration

$$\underline{\alpha}(n+1) = \underline{\alpha}(n) + a(n)\mathbf{G}(\underline{\alpha}), \tag{23}$$

where $\{a(n)\}$ are the usual stochastic approximation stepsize schedules. By the standard 'o.d.e. approach' to stochastic approximation [16], this tracks the asymptotic behavior of the ordinary differential equation (o.d.e.)

$$\dot{x}(t) = \mathbf{G}(x).$$

For a general network and coefficients $A_i, A_{i,j}, C_i$, the stability of the equilibrium points of the o.d.e. cannot be a priori assumed (see also [20] for this issue). However, for a special case where all the nodes are neighbors of each other, it can be shown that the (slightly modified) o.d.e. is globally asymptotically stable and hence the suggested iteration above is guaranteed to converge irrespective of the coefficients.

4.3 The Case of All Nodes Neighbor of Each Other

Consider the special case where for any node i, $\mathcal{N}(i) = \mathcal{N}$, i.e., all the nodes are neighbors of each other. This is a common scenario in wireless LANs spanning a small area (office etc.). The standard Slotted ALOHA system is yet another example of such scenario.

Recently, [1] has also considered a game theoretic approach to delay minimization in Slotted Aloha systems with the *retransmission probabilities* as decision variables. However, it does not consider the problem of nodes computing the optimal retransmission probabilities. The problem there also assumes symmetry, i.e., (unlike our case) all nodes have equal weightage thus resulting in equal optimal retransmission probabilities for each node.

For the present case where $S(i)$ is empty, it is seen that $P_s(i) = \Pi_{j\neq i}(1-\alpha_j)$, and $P_r(j,i) = \Pi_{k\neq j}(1-\alpha_k) = P_s(j)$. Note that $P_r(j,i) = P_r(j,k)$ for any $k, i \neq j$. The Kuhn-Tucker condition (Equation 20) can be written as

$$\sum_{j\neq i} A_{i,j}\frac{\alpha_j}{1-\alpha_j} = A_{i,i}, \quad \forall i.$$

Let $\beta_j := \frac{\alpha_j}{1-\alpha_j}$, $\zeta_{i,j} = A_{i,j},\ i \neq j$ and $\eta_i = A_{i,i}$. The above condition in matrix form is then,

$$\zeta \underline{\beta} = \underline{\eta}. \tag{24}$$

Remark: Equation 24 gives a complete characterization of the solution of optimization problem under consideration as a solution to a set of linear equations. This is a considerable simplification given the complex set of equations representing the optimization problem. Now we proceed to give a method to compute this optimum in a distributed manner.

4.4 The Algorithm

To solve Equation 24, the iteration to be considered is

$$\underline{\beta}(n+1) = \underline{\beta}(n) + a(n)(\underline{\eta} - \zeta \underline{\beta}(n)),$$

corresponding to the o.d.e. $\dot{x}(t) = (\underline{\eta} - \zeta x(t))$, whose stability, again, can not be apriori assumed. We thus consider the modified o.d.e. having same critical points

$$\dot{x}(t) = \zeta(\underline{\eta} - \zeta x(t)). \tag{25}$$

This will be stable if the matrix ζ is invertible, whence ζ^2 will be positive definite. The solution will correspond to the linear system

$$\zeta \underline{\eta} - \zeta^2 \underline{\beta} = 0, \tag{26}$$

which then has the same solution as (24). Thus node i will be solving the i^{th} row of (26). The iteration at node i is thus

$$\underline{\beta}(n+1) = \underline{\beta}(n) + a(n)\zeta(\underline{\eta} - \zeta \underline{\beta}(n)). \tag{27}$$

The algorithm run by the nodes based on the above iteration is detailed as follows.

1. Set the slot number $n = 0$. Initialize $\alpha_i^{(0)},\ 1 \leq i \leq N$, to some small positive values. Also let $N(i) = 0$, the last slot number when node i updated its attempt probability.
2. For $1 \leq i \leq N$, node i does the following operations:
   - It either decides to transmit with probability $\alpha_i^{(n)}$, or decides to receive with probability $1 - \alpha_i^{(n)}$.
   - If decided to transmit, a node sends the data packet destined for all the neighboring nodes. It also transmits the information relevant for other nodes for updating their attempt probabilities based on (27). In particular, node i transmits $\alpha_i^{(n)}$, $N(i)$, $\mathcal{N}(i)$.
   - If decided to sense the channel, do the following:

- If node i receives a signal that can be decoded correctly, then it checks if the received signal contains the update information. If yes, update node i's local information about its neighbors based on information extracted from the transmission received from its first hop neighbor. Here node i updates its estimate of $\alpha_j,\ j \neq i$, only if the value of $N(j)$ that it has now received is more than node i's copy of $N(j)$.
- Update $\alpha_i^{(n)}$ based on the updated information.
- Set $N(i) = n$.

3. $n = n + 1$, Go to step 2.

For numerical results based on an implementation of the algorithm, see [2].

5 Non-cooperative Forwarding in Ad Hoc Networks

As mentioned in the Introducion, in this work we consider a less aggressive punishment policy as an incentive for cooperation in ad hoc networks. We simply assume that if the fraction q' of packets forwarded by a mobile is less than the fraction q forwarded by other mobiles, then this will result in a decrease of the forwarding probability of the other mobiles to the value q'. We shall show that this will indeed lead to non-aggressive equilibria, yet not necessarily to complete cooperation. The reasons for adopting this milder punishment strategy are the following:

1. There has been criticism in the game-theoretical community on the use of aggressive punishments. For example, threats for aggressive punishments have been argued not to be credible threats when the punishing agent may itself loose at the punishing phase. This motivated equilibria based on more credible punishments known as subgame perfect equilibria [21].
2. An individual that adopts an "partially-cooperative" behavior (i.e. forwards packets with probability $0 < q < 1$) need not be considered as an "aggressive" individual, and thus the punishment needs not be "aggressive" either; it is *fair* to respond to such a partially-cooperative behavior with a partially-cooperative reaction, which gives rise to our mild punishment scheme.
3. The TFT policy would lead to complete cooperation at equilibrium. However, our milder punishment seems to us more descriptive of actual behavior in the society in which we do not obtain full cooperation at equilibrium (for example in the behavior of drivers on the road, in the rate of criminality etc.) It may indeed be expected that some degree of non-cooperative behavior by a small number of persons could result in larger and larger portions of the society to react by adopting such a behavior.

As already mentioned, incentive for cooperation in Ad-hoc networks have been studied in several papers, see [11,18,22,24]. Almost all previous papers however only considered utilities related to successful transmission of a mobile's packet to its neighbor. In practice, however, multihop routes may be required for a packet to reach its destination, so the utility corresponding to successful

transmission depends on the forwarding behavior of all mobiles along the path. The goal of our paper is therefore to study the forwarding taking into account the multihop topological characteristics of the path.

Most close to our work is the paper [11] which considers a model similar to ours (introduced in Section 5.1 below). [11] provides sufficient condition on the network topology under which each node employing the "aggressive" TFT punishment strategy results in a Nash equilibrium. In the present section, we show that a less aggressive punishment mechanism can also lead to a Nash equilibrium which has a desirable feature that it is less resource consuming in the sense that a node need not accept all the forwarding request.

5.1 The Model

Consider an Ad-hoc network described by a directed graph $G = (N, V)$. Along with that network, we consider a set of source-destination pairs O and a given routing between each source s and its corresponding destination d, of the form $\pi(s,d) = (s, n_1, n_2, \ldots, n_k, d)$, where $k = k(s,d)$ is the number of intermediate hops and $n_j = n_j(s,d)$ is the jth intermediate node on path $\pi(s,d)$. We assume that mobile j forwards packets (independently from the source of the packet) with a fixed probabilty γ_j. Let $\underline{\gamma}$ be the vector of forwarding probabilities of all mobiles. We assume however that each source s forwards its own packets with probability one. For a given path $\pi(s,d)$, the probability that a transmitted packet reaches its destination is thus:

$$p(s,d;\underline{\gamma}) = \prod_{j=1}^{k(s,d)} \gamma(n_j(s,d)).$$

If i belongs to a path $\pi(s,d)$ we write $i \in \pi(s,d)$. For a given path $\pi(s,d)$ of the form $(s, n_1, n_2, \ldots, n_k, d)$ and a given mobile $n_j \in \pi(s,d)$, define the set of intermediate nodes before n_j to be the set $S(s,d;n_j) = (n_1, ..., n_{j-1})$. The probability that some node $i \in \pi(s,d)$ receives a packet originating from s with d as its destination is then given by

$$p(s,d;i,\underline{\gamma}) = \prod_{j \in S(s,d;i)} \gamma(j).$$

Note that $p(s,d;d,\underline{\gamma}) = p(s,d;\underline{\gamma})$, the probability that node d receives a packet originating from source s and having d as its destination.

Define $O(i)$ to be all the paths in which a mobile i is an intermediate node. Let the rate at which source s creates packets for destination d be given by some constant λ_{sd}. Then the rate at which packets arrive at node i in order to be forwarded there is given by

$$\xi_i(\underline{\gamma}) = \sum_{\pi(s,d) \in O(i)} \lambda_{sd} p(s,d;i,\underline{\gamma}).$$

Let E_f be the total energy needed for forwarding a packet (which includes the energy for its reception and its transmission). Then the utility of mobile i that we consider is

$$U_i(\underline{\gamma}) = \sum_{n:(i,n)\in O} \lambda_{in} f_i(p(i,n;\underline{\gamma})) + \sum_{n:(n,i)\in O} \lambda_{ni} g_i(p(n,i;\underline{\gamma})) - aE_f\xi_i(\underline{\gamma}), \quad (28)$$

where f_i and g_i are utility functions that depend on the success probabilities associated with node i as a source and as a destination respectively and a is some multiplicative constant. We assume that $f_i(\cdot)$ and $g_i(\cdot)$ are nondecreasing concave in their arguments. The objective of mobile i is to choose γ_i that maximizes $U_i(\underline{\gamma})$. We remark here that similar utility function is also considered in [11] with the difference that node's utility does not include its reward as a destination, i.e., they assume that $g_i(\cdot) \equiv 0$.

Definition: *For any choices of strategy $\underline{\gamma}$ for all mobiles, define $(\gamma_i', \underline{\gamma}^{-i})$ to be the strategy obtained when only player i deviates from γ_i to γ_i' and other mobiles maintain their strategies fixed.*

In a noncooperative framework, the solution concept of the optimization problem faced by all players is the following:

Definition: *A Nash equilibrium, is some strategy set $\underline{\gamma}^*$ for all mobiles such that for each mobile i,*

$$U_i(\underline{\gamma}^*) = \max_{\gamma_i'} U_i(\gamma_i', (\underline{\gamma}^*)^{-i}).$$

We call $argmax_{\gamma_i'} U_i(\gamma_i', \underline{\gamma}^{-i})$ the set of optimal responses of player i against other mobiles policy $\underline{\gamma}^{-i}$ (it may be an empty set or have several elements).

In our setting, it is easy to see that for each mobile i and each fixed strategy $\underline{\gamma}^{-i}$ for other players, the best response of mobile i is $\gamma_i = 0$ (unless $O(i) = \emptyset$ in which case, the best response is the whole interval $[0, 1]$). Thus the only possible equilibrium is that of $\gamma_i = 0$ for all i. To overcome this problem, we consider the following "punishing mechanism". in order to incite mobiles to cooperate.

Definition: *Consider a given set of policies $\underline{\gamma} = (\gamma, \gamma, \gamma, ...)$. If some mobile deviates and uses some $\gamma' < \gamma$, we define the punishing policy $\kappa(\gamma', \gamma)$ as the policy in which all mobiles decrease their forwarding probability to γ'.*

When this punishing mechanism is enforced, then the best strategy of a mobile i when all other mobiles use strategy γ is γ' that achieves

$$J(\gamma) := \max_{\gamma' \leq \gamma} U_i(\underline{\gamma}') \quad (29)$$

where $\underline{\gamma}' = (\gamma', \gamma', \gamma', ...)$.

Definition: *If some γ^* achieves the minimum in (29) we call the vector $\underline{\gamma}^* = (\gamma^*, \gamma^*, \gamma^*, ...)$ the equilibrium strategy (for the forwarding problem) under threats. $J(\gamma)$ is called the corresponding value.*

Remark: Note that $\gamma^* = 0$ is still a Nash equilibrium.

5.2 Utilities for Symmetrical Topologies

By symmetrical topology we mean the case where f_i, g_i and ξ_i are independent of i. This implies that for any source-destination pair (s, d), there are two nodes s' and d' such that the source-destination pairs (s', s) and (d, d') are identical to (s, d) in the sense that there view of the network is similar to that of (s, d). This implies that, under the punishment mechanism where all nodes have same forwarding probability, we have $p(s, d; \underline{\gamma}) = p(s', s; \underline{\gamma})$. Thus we can replace the rewards $f_i + g_i$ by another function that we denote $f(\cdot)$.

Consider $\underline{\gamma}$ where all entries are the same and equal to γ, except for that of mobile i. For a path $\pi(s, d)$ containing n intermediate nodes, we have $p(s, d; \underline{\gamma}) = \gamma^n$. Also, if a mobile i is $n + 1$ hops away from a source, $n = 1, 2, 3, ...,$ and is on the path from this source to a destination (but is not itself the destination), then $p(s, d; i, \underline{\gamma}) = \gamma^n$. We call the source an "effective source" for forwarding to mobile i since it potentially has packets to be forwarded by mobile i. Let $h(n)$ be the rate at which all effective sources located $n + 1$ hops away from mobile i transmit packets that should use mobile i for forwarding (we assume that h is the same for all nodes). Let $\lambda^{(n)}$ denote the rate at which a source s creates packets to all destinations that are $n + 1$ hops away from it. Then we have

$$U_i(\underline{\gamma}) = \sum_{n=1}^{\infty} \lambda^{(n)} f(\gamma^n) - aE_f \sum_{n=1}^{\infty} h(n)\gamma^n. \tag{30}$$

The equilibrium strategy under threat is then the value of γ that maximizes the r.h.s.

Remark: If we denote by $\Lambda(z) = \sum_{n=1}^{\infty} z^n \lambda^{(n)}$ the generating function of $\lambda^{(n)}$ and $H(z) := \sum_{n=1}^{\infty} z^n h(n)$ the generating function of h. Then

$$\max_{\gamma} \Big(\Lambda(\gamma) - aE_f H(\gamma) \Big)$$

is the value of the problem with threats in the case that f is the identity function.

5.3 Examples

In this section we present, by means of two examples, the effect of imposing the proposed punishment mechanism.

5.4 An Asymmetric Network

Consider the network shown in Figure 1. For this case nodes 1 and 4 have no traffic to forward. Note also that if we assume that $g_3(\cdot) \equiv 0$ in Equation 28 then node 3 has no incentive even to invoke the punishment mechanism for node 2. This will result in no cooperation in the network. Assume for the time being that $f_2(x) = g_3(x) = x$, i.e., f_2 and g_3 are identity functions. In this case it is seen that the utility functions for nodes 2 and 3 are, assuming $\lambda_{13} = \lambda_{24} = 1$, $U_2(\gamma_2, \gamma_3) = \gamma_3 - aE_f\gamma_2$ and $U_3(\gamma_2, \gamma_3) = \gamma_2 - aE_f\gamma_3$. When we impose the

punishment mechanism, it turns out that the equilibrium strategy for the two nodes is to always cooperate, i.e., $\gamma_2 = \gamma_3$. This is to be compared with the TFT strategy of [11] which would imply $\gamma_2 = \gamma_3 = 0$.

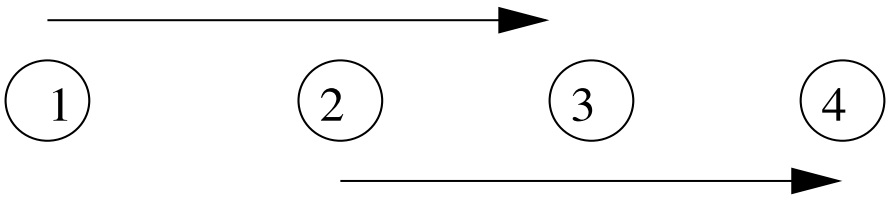

Fig. 1. An asymmetric network

5.5 A Symmetric Network: Circular Network with Fixed Length of Paths

We consider here equally spaced mobile nodes on a circle and assume that each node i is a source of traffic to a node located L hops to the right, i.e. to the node $i + L$.

Let the rate of traffic generated from a source be λ. For this case, $h(n) = \lambda I_{\{n \leq L-1\}}$. Also, $\lambda^{(n)} = \lambda I_{\{n=L\}}$, for some λ. It follows from Equation 30 that the utility function for mobile i is

$$U_i(\underline{\gamma}) = \lambda f(\gamma^{L-1}) - aE_f \lambda \sum_{n=0}^{L-2} \gamma^n.$$

For $f(\cdot)$ an identity function, we see that $U_i(\underline{\gamma}) = \lambda \left[\gamma^{L-1} - aE_f(\gamma^{L-2} + \gamma^{L-3} + \ldots + \gamma + 1)\right]$. Note that if $L = 2$ and $a = \frac{1}{E_f}$, the utility function is independent of γ hence in this case the equilibrium strategy is any value of forwarding probability. Also, if $aE_f \geq 1$, the equilibrium strategy is $\gamma = 0$. We will have more to say on this in th next section where we study the structure of equilibrium strategy for symmetric network.

5.6 Algorithm for Computing the Equilibrium Strategy in a Distributed Manner

It is interesting to design distributed algorithms which can be used by the mobiles to compute the equilibrium strategy and simultaneously enforce the proposed punishment mechanism. The obvious desirable features of such an algorithm are that it should be decentralised, distributed scalability and should be able to adapt to changes in network.

We propose such an algorithm in this section. We present it, for ease of notation, for the case of symmetric network. Assume for the moment that $f(\cdot)$ is the identity function. In this case each node has to solve the equation (recall the notation of Section 5.2)

$$U'(\gamma) = \Lambda'(\gamma) - KH'(\gamma) = 0, \tag{31}$$

where the primes denote the derivatives with respect to γ. In general this equation will be nontrivial to solve directly. For the case of more general network, one needs to compute the derivative of the utility function of Equation 28, the rest of procedure that follows is similar.

Note that in the above expression we first assume that the forwarding probabilities of all the nodes in the network are same (say γ) and then compute the derivative with respect to this common γ. This is because in the node must take the effect of punishment mechanism into account while computing its own optimal forwarding probability, i.e., a node should assume that all the other nodes will use the same forwarding probability that it computes.

Thus, solving Equation 31 is reduced to a single variable optimization problem. Since the actual problem from which we get Equation 31 is a maximization problem, a node does a gradient *ascent* to compute its optimal forwarding probability. Thus, in its n^{th} computation, a node i uses the iteration

$$\gamma_i^{(n+1)} = \gamma_i^{(n)} + a(n)(\Lambda'(\gamma_i^{(n)}) - KH'(\gamma_i^{(n)})), \tag{32}$$

where $a(n)$ is a sequence of learning parameters in stochastic approximation algorithms.

The relation to stochastic approximation algorithm here is seen as follows: the network topology can be randomly changing with time owing to node failures/mobility et cetera. Thus a node needs to appropriately modify the functions $\Lambda(\cdot)$ and $H(\cdot)$ based on its most recent view of the network (this dependence of $\Lambda(\cdot)$ and $H(\cdot)$ on n is suppressed in the above expression).

It is a matter of choice when a node should update its estimate of its forwarding probability, i.e., does the computations mentioned above. One possibility, that we use, is to invoke the above iteration whenever the node receives a packet that is meant for it.

Though the above is a simple stochastic approximation algorithm, it requires a node to know the topology of the part of network around itself. This information is actually trivially available to a node since it can extract the required information from the packets requesting forwarding or using a neighbour discovery mechanism. However, in case of any change in the network, there will typically be some delay till a node completely recognizes the change. This transient error in a node's knowledge about the network whenever the network changes is ensured to die out ultimately owing to the assumption of finite second moment for the learning parameters.

It is known by the o.d.e. approach to stochastic approximation algorithm that the above algorithm will asymptotically track the o.d.e. [15]:

$$\dot{\gamma}_i(t) = \Lambda'(\gamma_i(t)) - KH'(\gamma_i(t)), \tag{33}$$

and will converge to one of the *stable* critical points of o.d.e. of Equation 33. It is easily seen that a local maximum of the utility function forms a stable critical point of Equation 33 while any local minimum forms an unstable critical point. Thus the above algorithm inherently makes the system converge to a local maximum and avoids a local minimum.

However, it is possible that different nodes settle to different local maxima (we have already seen that there can be multiple maxima). The imposed punishment mechanism then ensures that all the nodes settle to the one which corresponds to the lowest values of γ. This is a desirable feature of the algorithm that it inherently avoids multiple simultaneous operating points. An implementation of the punishment mechanism is described next.

5.7 Distributed Implementation of the Punishment Mechanism

An implementation of punishment mechanism proposed in Section 5.1 requires, in general, a node to know about the misbehaving node in the network, if any. Here we propose a simple implementation of the punishment mechanism which requires only local information for its implementation.

Let $\mathcal{N}(i)$ be the set of neighbours of node i. Every node computes its forwarding policy in a distrubuted manner using the above mentioned stochastic approximation algorithm. However, as soon as a neighboring node is detected to misbehave by a node, the node computes its forwarding policy as follows:

$$\gamma_i^* = \min\{\gamma_i, \min_{j \in \mathcal{N}(i)} \hat{\gamma}_j\} \tag{34}$$

where γ_i and $\hat{\gamma}_j$ represents, respectively, the forwarding policy adopted by node i and the estimate of node j's forwarding probability available to node i. γ_i^* represents the new policy selected by node i. Note here that γ_i is still computed using iteration of Equation 32. We are also assuming here that a node can differentiate between a misbehaving neighbouring node and the failure/mobility of a neighbouring node.

This punishment propagates in the network until all the nodes in the network settle to the common forwarding probability (corresponding to that of the misbehaving node). In particular, the effect of this punishment will be seen by the misbehaving ndoe as a degradation in its own utility. Suppose now that the misbehaving node, say n_i, decides to change to a cooperative behavior: at that point, it will detect and punish its neighbors because of the propagation of the punishment that induced its neighbouring nodes to decrease their forwarding policy. Thus, the intial punishment introduces a negative loop and the forwarding policy of every node of the network collapses to the forwarding policy selected by the misbehaving node. Since now every node in the network has same value of forwarding probability, none of the nodes will be able to increase its forwarding probability even if none of the node is misbehaving now.

An example of this phenomenon can be seen from the network of Figure 1. Assume that $\gamma_2 = \gamma_3 = \gamma$ and now node 2 reduces γ_2 to a smaller value γ'. Owing to the punishment mechanism, node 3 will respond with $\gamma_3 = \gamma'$. This

will result in a reduced utility for node 2 which would then like to increase γ_2. But, since $\gamma_3 = \gamma'$, the punishing mechanism would imply that $\gamma_2 = \gamma'$ as well. This *lock-in* problem is avoided by the solution proposed below.

We modify our algorithm to account for the above mentioned effect. Our solution is based on timers of a fixed duration. When a node enters in the punishing phase (starts punishing some of its neighbour) the local timer for that node is set and the forwarding policy is selected as in equation 34. When the timer expires, the punishing node evaluates its forwarding policy as if there were no misbehaving nodes, then uses some of standard mechanism to detect any persistent misbehavior (this also helps distinguishing between a misbehaving node and a failed/moved node). In the case no misbehaviors are detected, depending on the choice of the learning parameter of the stochastic apporximation algotithm, the forwarding policy of the network eventually returns to the optimal value for the network. If the neighboring node continues to misbehave, the timer is set again and the punishment mechanism is re-iterated. We assume that the sequence of learning parameters by a node is restarted each time the timer is set.

Remark: It is interesting to see that the proposed implementation of the punishing mechanism is actually having a storage complexity for a node that grows only with the number of its neighbouring nodes (Equation 34). Computational complexity is also not large as it depends only on the distance (hops) from a node to its farthest destination (Equation 32).

See [3] for numerical results and discussions on the implementation of the algorithm.

6 Conclusion

We have proposed and analysed a distributed scheme for adapting the random access probabilities in a wireless ad hoc network and tested it on some simple scenarios. The advantage of the scheme is its simplicity, made possible by a simplified model and a judicious choice of the performance measures. This makes it attractive from an implementation point of view.

We use the framework of non-cooperative game theory to provide incentives for cooperation in the case of wireless Ad-hoc networks. The incentive proposed in the paper is based on a simple punishment mechanism that can be implemented in a completely distributed manner with very small computational complexity. The advantage of the proposed strategy is that it results in a less "aggressive" equilibrium in the sense that it does not result in a degenerate scenario where a node either forwards all the requested traffic or does not forward any of the request.

Acknowledgements The work of the E. Altman, V.S. Borkar and A.A. Kherani was supported in part by project no. 2900-IT-1 from the *Centre Franco-Indien pour la Promotion de la Recherche Avancee* (CEFIPRA). The work of E. Altman was partially supported by the European Network of Excellence EURO NGI.

References

1. E. Altman, R. El Azouzi and T. Jimenez: Slotted Aloha as a Stochastic Game with Partial Information, WiOpt'03, Sophia Antipolis, France, (2003).
2. E. Altman, V. S. Borkar and A. A. Kherani: Optimal Random Access in Networks with Two-Way Traffic, in PIMRC 2004, Spain.
3. E. Altman, A. A. Kherani, P. Michiardi and R. Molva: Non-cooperative Forwarding in Ad-hoc Networks, INRIA Report No. RR-5116, Sophia-Antipolis, France, February 2004.
4. Bertsekas, D., Gallager, R.: Data Networks. Prentice Hall, (1992).
5. Bharghavan, V., Demers, A., Shenker, S., Zhang, L.: MACAW: A Media Access Protocol for Wireless LANs. ACM SIGCOMM, (1994).
6. Borkar, V. S.: Asynchronous Stochastic Approximation. SIAM Journ. of Control and Optimization, **36** (1998).
7. V. S. Borkar and A. A. Kherani: Random Access in Wireless Ad Hoc Networks as a Distributed Game, in proceedings of WiOpt 2004, Cambridge.
8. Cali, F., Conti, M., Gregori, E.: Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit. IEEE/ACM Transactions on Networking, (2000).
9. J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring. Modelling incentives for collaboration in mobile Ad-hoc networks. In *Proceedings of WiOpt'03*, Sophia-Antipolis, France, 3-5, March 2003.
10. D. Dutta, A. Goel and J. Heidemann, "Oblivious AQM and Nash Equilibria", IEEE Infocom, 2003.
11. M. Félegyházi, L. Buttyán and J. P. Hubaux, "Equilibrium analysis of packet forwarding strategies in wireless Ad-hoc entworks – the static case", PWC 2003 Personal Wireless Communications, Sept. 2003, Venice, Italy.
12. IEEE Computer Society LAN MAN Standards Committee.: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Standard 802.11-1997, (1997).
13. Johnson, D., Maltz, D., Broch, J.: DSR: The Dynamic Source Routing Protocol for Multi Hop Wireless Ad Hoc Networks. Ad Hoc Networking, Addision-Wesley, (2001).
14. Karnik, A., Kumar, A.: Optimal Self-Organization of Wireless Sensor Networks. Infocom 2004.
15. H. J. Kushner and G. Yin, "Stochastic Approximation Algorithms and Applications," Springer-Verlag, 1997.
16. Kushner, H. J., Yin, G.: Stochastic Approximation Algorithms and Applications. Springer-Verlag, (1997).
17. A. Mas-Colell, M. D. Whinston and J. R. Green: Microeconomic Theory, Oxford Univ. Press, (1995).
18. P. Michiardi and R. Molva. A game theoretical approach to evaluate cooperation enforcement mechanisms in mobile Ad-hoc networks. In *Proceedings of WiOpt'03*, Sophia-Antipolis, France, 3-5, March 2003.
19. Perkins, C. E., Bhagwat, P.: Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. Computer Communications Review, (1994).
20. J. B. Rosen: Existence and Uniqueness of Equlibrium Points for Concave N-Person Games, Econometrica, (1965).

21. L. Samuelson, "Subgame Perfection: An Introduction," in John Creedy, Jeff Borland and Jürgen Eichberger, eds., Recent Developments in Game Theory, Edgar Elgar Publishing, 1992, 1-42.
22. V. Srinivasan, P. Nuggehalli, C. F. Chiasserini and R. R. Rao, "Cooperation in wireless Ad-hoc networks", *Proceedings of IEEE Infocom*, 2003.
23. Sundaram, R. K.: A First Course in Optimization Theory. Cambridge University Press, (1999).
24. A. Urpi, M. Bonuccelli, and S. Giordano. Modelinig cooperation in mobile Ad-hoc networks: a formal description of selfishness. In *Proceedings of WiOpt'03*, Sophia-Antipolis, France, 3-5, March 2003.