Self-optimizing Routing in MANETs with Multi-class Flows

Pierre Coucheney, Bruno Gaujal and Corinne Touati
INRIA Rhône-Alpes and LIG, MESCAL project, Grenoble France,
{pierre.coucheney, bruno.gaujal, corinne.touati}@inria.fr

Abstract—**In this paper we show how game theory and Gibbs sampling techniques can be used to design a self-optimizing algorithm for minimizing end-to-end delays for all flows in a multi-class mobile ad hoc network (MANET).**

This is an improvement over the famed Ad-Hoc On-demand Distance Vector (AODV) protocol, that computes the routes with minimal number of hops for each flow in a multi-flow ad-hoc network. Here, the load of each flow is taken into account to choose the best route (in terms of delays) among a fixed number of routes. The algorithm can be implemented in a fully distributed and asynchronous way and is guaranteed to converge to the global optimal configuration. Numerous numerical experiments show that the gain over AODV, computed over a large number of networks, is quite substantial.

I. Introduction

Mobile ad hoc networks (MANET) are collection of geographically distributed nodes that can be self-configured to form a network without predetermined topology (see [1]). Significant research has been directed towards implementing application dependent Quality of Service (QoS) requirements and has addressed power control, coding, adaptive techniques at the link layer, scheduling in the Medium Access Control (MAC) layer, and energy and delay constrained routing in the network layer. In MANETs, it is important to find and maintain correct routes to the destination in a changing topology resulting from node failure or mobility. Different routing protocols use one or more metrics to determine optimal routes. The most widely used routing protocols are the Ad-hoc On-demand Distance Vector (AODV [2]), the Dynamic Source Routing (DSR), the Destination Sequenced Distance Vector (DSDV) and the Temporally-Ordered Routing Algorithm (TORA). All these routing protocols use the shortest-hop metric to choose the best route. Consequently, they do not take into account‘ the load on each link. Particularly there may be competition between users to access to the best path in term of delay. It is often noticed that, without coordination between users, this may lead to inefficient use of the network resources by increasing the congestion of the network.

Here, we consider a wireless ad hoc network in which several users send data from sources to destinations of the network. Users may belong to several classes of traffic[1], depending on their critically. The delay experienced by a user on a path is the sum of the delay on each link of this path. The delay on a link is a function of the load on this link, *i.e.* the arrival rate at this link. The delay also depends on the priority of the stream. For example, users with delay constraints may be consider as higher priority flows than users without delay constraint (for example VoIP vs FTP streams). So, each user may select one path to connect the source to the destination in a set of available paths (at the initialization of AODV protocol, a set of paths is stored in memory). As an extension of the protocol AODV that selects the shortest path in term hop, we aim at associate each user to one of her set of path in order to minimize the overall delay in the network.

The main contribution of this paper is to provide a fully distributed algorithm, based on the protocol AODV, for which we can analytically prove that it reaches an optimal routing. The algorithm, similarly to the original AODV is based on the current state of the system and reacts to change in the topology or the set of active connections. We provide extensive simulations of the algorithm in several scenarios. Also, we propose a simple heuristic of this algorithm that achieve good performances, and that is more reactive to the dynamism of the system than the optimal algorithm. The theoretical validation of the algorithm is based on three fields: game theory, Gibbs sampling and distributed algorithms.

a) Game theory: Our system can be seen as a multi-commodity (several source destination) weighted (each user has a specific rate) congestion game. Therefore, there is no guarantee of existence of a pure Nash equilibrium, *i.e.* an association such that no user has incentive to change her route. As a consequence, selfish dynamics such that best response dynamics may not converge, hence resulting in perpetual change of route. The first contribution of this paper is to provide a pricing mechanism, based on measurements on each link, that transforms this game into a potential game [4], such that the potential function is the global delay, *i.e.* the function to maximize. The benefit is that the local behavior of users is aggregated in this real valued function, called the potential.

b) Gibbs sampler: There exists classical dynamics used in game theory (*e.g.* best response, replicator dynamics) that may converge to equilibria of potential games (*i.e.* local optimizers). But in our general setting, there is no guarantee of convergence speed of these dynamics [5], and more importantly, of global optimality. On the other hand, there exists an algorithm based on Gibbs samplers and on a specific cooling schedule, that is proved, under some conditions, to reach a globally optimal point [6], [7].

c) Distributed computation: We show that both this algorithm and the pricing mechanism can be implemented in

The authors would like to thank Prof. Jie Li and Khoriba Ghada, University of Tsukuba, Japan for insightful discussions.

This work was performed as part of the INRIA - Alcatel Lucent Bell Labs common Laboratory.

[1]For example, there are four priority classes in UMTS: two real-time (voice and streaming) and two non-real time (interactive and background) [3].

a fully distributed and asynchronous way, where communication is only possible between neighbors. Furthermore, the algorithm takes advantage of the message passing protocol at the MAC layer so that its impact over the normal behavior of the ad-hoc network is almost negligible.

II. Model

A. Optimization Problem

We consider an ad-hoc network represented by an oriented graph, whose nodes are the machines that transmit or receive packet flows. We denote by $\mathcal{N}$ the set of nodes. There is an arc between two nodes if the corresponding machines can communicate. Since we consider wireless ad-hoc network, the communication is not necessarily symmetric (reception capacity depends on antenna quality while emission range depends essentially on power and battery capacity), hence the graph may not have arcs in both directions for any two nodes. On this network, a set of *users* (also called *flows* in the following) $\mathcal{U}$ seek to transmit some flow of packets from a source node to a destination node, and we assume that the flow of packets has a stationary rate. Each user belongs to a priority class (from 1 to C) and the network associated a weight to each class. The weight can be seen as a relative measure of the importance of users of a given class. Of course, the higher the priority, the larger the weight. Each user u is therefore equipped with a weight w_u depending on its class.

For each user $u \in \mathcal{U}$, we fix Π_u, a set of simple *paths* that correspond to the routes that user u can select. A path is a sequence of machines that connect the source to the destination. Then, we denote by $s_u \in \Pi_u$ the current choice of user u. $s = (s_u)_{u\in\mathcal{U}}$ is called an *association* of each user to a path, and $\mathcal{S}$ is the set of all possible associations.

Let ℓ_n be the state of node n: it is a binary vector $(\ell_n^u)_{u\in\mathcal{U}}$ such that $\ell_n^u = 1$ if user u uses the node n, *i.e.* $n \in s_u$. Finally, the expected delay in node n for packets from user u is denoted $d_n^u(\ell_n)$ and the delay of the whole path $p \in \Pi_u$ for user u is $D_p^u(s)$. Here we have the following features:

- The delay of a flow over a given route only comes from delays on nodes (the transmission delays from one node to the next are neglected).
- The delays are additive on paths, *i.e.* $D_p^u(s) = \sum_{n\in p} d_n^u(\ell_n)$.
- The delays over one node are not the same for all users: one may have $d_n^u(\ell_n) \neq d_n^v(\ell_n)$, for any two users u and v using node n as long as users u and v belong to different priority classes. If user u has priority over user v, her delay on node n may be shorter. Fixed preemptive priorities of flows are used in the numerical experiments given in Section V.

The problem is, at each connection arrival or departure, to find an association of each user to one of its available paths in order to minimize the global delay. More formally, we seek to solve the following problem:

$$\min_{s\in\mathcal{S}} F(s), \tag{1}$$

where $F(s) \overset{\text{def}}{=} \sum_{u\in\mathcal{U}} w_u D_{s_u}^u(s) = \sum_{u\in\mathcal{U}} \sum_{n\in s_u} w_u d_n^u(\ell_n)$.

B. Potential Games

Solving Problem (1) can be done by exploring all the the possible associations, but this centralized brute force approach has a high combinatorial complexity that make it unpractical for large systems. Another alternative is to exploit the properties of the problem, if any, like convexity [8]. However, these solutions also need a centralized controller. Furthermore, convexity does not hold here because of interferences. Therefore, it is not suitable in ad-hoc networks.

The approach taken here, is based on a self-optimizing principle. Each user finds the solution for an individual cost, instead of the global cost. Here, minimizing the individual delays will not lead to a global optimization algorithm because of dependencies between individual delays.

However, in some cases, individual optimizations provide a global optimal configuration. This is typically the case with potential games, which are games such that the individual difference of delays by unilaterally changing one path is exactly the difference of the global cost. More formally, we say that the game $(\mathcal{U}, (\Pi_u)_u, D)$ is an *exact potential game* (called potential game in the sequel) if there exists a potential function G (not necessarily unique) such that:

$$\forall u \in \mathcal{U}, \forall p \in \Pi_u, D_{s_u}^u(s) - D_{s_u}^u(p, s_{-u}) = G(s) - G(p, s_{-u}), \tag{2}$$

where s_{-u} is the classical notation that refers to the choice of all users except u in the association s.

Potential games were first defined in the famous paper [4], whereas the potential argument had been previously used in [9] to show the existence of a pure Nash equilibrium in congestion games. A classical result says that, in potential games, the local minimizers of the potential are Nash equilibriums of the game. Hence the existence of a potential ensures the existence of a Nash equilibrium, which is not true in general (if we only consider *pure* strategies for the game). Here the natural game (using the delays as costs) is not a potental game and a Nash equilibrium may not exist.

In the following, we will show how the function to optimize in Problem (1) can be transformed into a potential function of a new game. This is done by introducing shadow prices (called repercussion delays here, because they are based on the impact of user u over the delays of others). Consequently, if each user minimizes her repercussion delay, then the global cost function F (sum of the actual delays) will be minimized.

Definition 1: The repercussion delay on each node is:

$$\delta_n^u(\ell_n) \overset{\text{def}}{=} w_u d_n^u(\ell_n) - \sum_{\substack{v\neq u:\\ s_v \ni n}} \left(w_v d_n^v(\ell_n - e_n^u) - w_v d_n^v(\ell_n)\right). \tag{3}$$

As before, the repercussion delay for u on path p is $\Delta_p^u(s) = \sum_{n\in p} \delta_n^u(\ell_n)$.

Proposition 1: The game with repercussion delays (3), is a potential game, whose potential is $F(.)$ defined in Problem (1).

Proof: According to the definition given by Equation (2), we just have to show that $\forall u \in \mathcal{U}, \forall s_{-u}, \forall p, q \in \Pi_u, \Delta_p^u(p, s_{-u}) - \Delta_q^u(q, s_{-u}) = F(p, s_{-u}) - F(q, s_{-u})$. For this, let us rewrite $F(s) = \sum_{n\in\mathcal{A}} \sum_{u:\, s_u \ni n} w_u d_n^u(\ell_n)$. Then,

$$\begin{aligned}
&F(p, s_{-u}) - F(q, s_{-u})\\
&= \sum_{n\in p\backslash q} w_u d_n^u(\ell_n) + \sum_{n\in p\backslash q} \sum_{\substack{v\neq u:\\ s_v\ni n}} (w_v d_n^v(\ell_n) - w_v d_n^v(\ell_n - e_n^u))\\
&- \sum_{n\in q\backslash p} w^u d_n^u(\ell_n) - \sum_{n\in q\backslash p} \sum_{\substack{v\neq u:\\ s_v\ni n}} (w_v d_n^v(\ell_n - e_n^u) - w_v d_n^v(\ell_n))\\
&= \sum_{n\in p\backslash q} \delta_n^u(\ell_n) - \sum_{n\in q\backslash p} \delta_n^u(\ell_n)\\
&= \Delta_p^u(p, s_{-u}) - \Delta_q^u(q, s_{-u}).
\end{aligned}$$

■

As a consequence of the proposition, the best response algorithm as defined in Algorithm 1 will converge to a locally optimal association that is a Nash equilibrium of the game, *i.e.* an association such that no user has incentive to change. Conversely, every Nash equilibrium is a local minimum for the potential and can be obtained using the best response algorithm. The reason is that, according to this dynamic, the potential is strictly increasing while a user has incentive to change her path. As the number of associations is finite, the best response algorithm will reach such an association in finite time.

Algorithm 1 Best response algorithm.

```
while a user has incentive to change do
    Choose randomly a user $u \in \mathcal{U}$
    forall path $p \in \Pi_u$ do
        compute the value of the repercussion delay $\Delta_p^u(s)$
    Choose $p$ that minimizes $\Delta_p^u(s)$
```

The main advantage of best response algorithm is that it can be implemented in a fully distributed way and that it usually converges extremely fast. But its main drawback is that it may not converge to a globally optimal association. In many cases, however, one can bound the ratio between the worst Nash equilibrium (a possible association found by best response) and the global optimal. This ratio is known as the *price of anarchy* [10] when the function to optimize is the social cost. Here, we are looking at the ratio between the optimal social cost for the delays (the function we seek to optimize) and the cost of the worst Nash equilibrium for the game with *repercussion* delays [2]. Since, it does not exactly correspond to the concept of the price of anarchy, the methods to bound the value of the price of anarchy may be adapted in our case. Some bounds on the price of anarchy are known in the case of non-atomic games (where each user has a negligible impact on the system), and for polynomial cost function [11]. For the case of atomic games (as it is the case here), or even a mix between atomic and non-atomic games, some recent works [12] provide bounds on the price of anarchy. However these results do not apply here since the derivative of the delay function is not bounded (due to the limited capacity of each link). Here, a Nash equilibrium can be arbitrarily far from the optimal cost. Therefore the best response approach may reach a local minimum arbitrarily far from the optimal association. This is illustrated in the following example.

[2]Note that the best Nash equilibrium is the optimal association, due to the potential property.

Consider the ad-hoc network given in Figure 1. Assume that there are 2 users in the same priority class sending some flow from source A to destination B, and from source C to destination D respectively, at the same rate $1\,kb/s$. Denote by u and v the users. We define a path as the sequence of nodes used by the path. Here, $\Pi_u = \{(U, V), (X, Y)\}$ and $\Pi_v = \{(U, Y), (X, V)\}$. As in the Section V, we assume the delay on each node to be given by the mean delay of a $M/M/1$ queue. The service rates are given by $2+2\varepsilon, 2+\varepsilon, 3, 2+\varepsilon\, kb/s$ for, respectively, nodes X, Y, U, V. The matrix of repercussion delays is given in Table I. The user 1 (resp. 2) chooses the row (resp. column), and her delay is given by the first (resp. second) component of the pair. So, one can check that there may be 2 pure Nash equilibria in the game with repercussion utilities namely $NE_1 = ((U, V), (U, Y))$ and $NE_2 = ((XY), (XV))$. Then, the global delay (1) for these equilibrium is $F(NE_1) = 4 + \frac{2}{1+\varepsilon}\, s$ and $F(NE_2) = \frac{2}{\varepsilon} + \frac{2}{1+\varepsilon}\, s$. Hence $\frac{F(NE_2)}{F(NE_1)} \to \infty$ as $\varepsilon \to 0$. Consequently, if the initial association is chosen uniformly between the four possibilities, and if the first user to change its path is also chosen uniformly, then the probability that the best response algorithm selects an equilibrium that is arbitrarily worse than the optimal one, is 0.5.

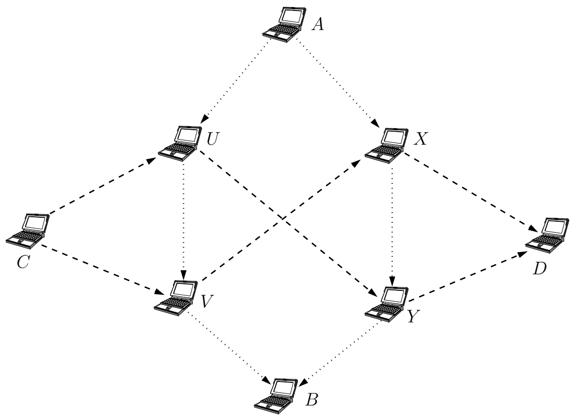

Figure 1: Ad-hoc network with unbounded performance ratio between the worst equilibrium (in the game with repercussions) and the best equilibrium for the global cost.

However, in the following section, we present an algorithm that is proved to asymptotically converges to a globally optimal association.

III. Optimal Association Using Gibbs Sampling

In this section, we describe an optimal algorithm in discrete time that associate each user to a path, in order to minimize

Table I: Matrix of repercussion delays

	$(U\ ,\ Y)$	$(X\ ,\ V)$
(U,V)	$\left(3.5+\frac{1}{1+\varepsilon}\ ,\ 3.5+\frac{1}{1+\varepsilon}\right)$	$\left(0.5-\frac{1}{1+\varepsilon}+\frac{4}{\varepsilon}\ ,\ \frac{1}{1+2\varepsilon}-\frac{1}{1+\varepsilon}+\frac{4}{\varepsilon}\right)$
(X,Y)	$\left(\frac{1}{1+2\varepsilon}-\frac{1}{1+\varepsilon}+\frac{4}{\varepsilon}\ ,\ 0.5-\frac{1}{1+\varepsilon}+\frac{4}{\varepsilon}\right)$	$\left(\frac{1}{1+\varepsilon}-\frac{1}{1+2\varepsilon}+\frac{2}{\varepsilon}\ ,\ \frac{1}{1+\varepsilon}-\frac{1}{1+2\varepsilon}+\frac{2}{\varepsilon}\right)$

the global delay. This algorithm is based on a Gibbs sampling technique.

Algorithm 2 Optimal association algorithm using Gibbs sampling

```
forall time epoch $t$ do
    Choose randomly a user $u \in \mathcal{U}$;
    forall path $p \in \Pi_u$ do
        Compute the repercussion delay $\Delta_p^u(s)$;
        Set the temperature to $T := -\ln(t)$;
        $r_p^u := \exp(\Delta_p^u(s)T)$;
    Pick path $p \in \Pi_u$ with probability $\frac{r_p^u}{\sum_{q \in \Pi_u} r_q^u}$;
```

Theorem 1: The association computed by Algorithm 2 asymptotically converges to a global optimal association.

Proof: Let $S(t)$ be the association of users to path at the t^{th} iteration of the algorithm. When the temperature T is fixed, one can see that $S(t)$ is a homogeneous Markovian process whose transition probability (that does not depend on n) is $\mathbb{P}[S(n+1) = (p, s_{-u}) | S(n) = (s_u, s_{-u})] = \frac{1}{|\mathcal{U}|} \frac{\exp(\Delta_p^u(s)T)}{\sum_{q \in \Pi_u} \exp(\Delta_q^u(s)T)} = \frac{1}{|\mathcal{U}|} \frac{\exp(F(p, s_{-u}))}{\sum_{q \in \Pi_u} \exp(F(q, s_{-u}))}$, by the potential property. It follows from a direct computation that the distribution that has weight $\frac{\exp(F(s))}{\sum_{s'} \exp(F(s'))}$ at association s is a stationary distribution of the Markov chain. Since the chain is irreducible and aperiodic, the probability to be in association s is asymptotically given by the stationary distribution. It follows that $\lim_{T \to 0} \lim_{t \to \infty} S(t) \in \operatorname{argmin} F(s)$.

Inverting the limits in the previous equation is valid when the temperature T decreases as the inverse of the logarithm of time t. In this case, the chain is non homogeneous, but using Theorem 8.1 in [13] one can show that the chain is strongly ergodic. Therefore, it converges for every initial distribution to the stationary distribution. The only possible limiting distribution is uniform on the associations of minimal global delay, and 0 elsewhere. ■

When applying the Gibbs sampling algorithm to the example given in Figure 1, with $\varepsilon = 0.1$, one gets the behavior given in Figure 2. The Gibbs sampling algorithms visits the bad local minimum a few times when the temperature is still high (and leaves it quite fast). When the temperature cools, the bad equilibrium is never visited again, so that the probability that the association computed by the Gibbs sampler is the bad Nash equilibrium goes to 0 very fast (compared with 1/2 for best response).

A question that naturally arises is whether this algorithm reaches optimal associations in more general settings. In [14], the authors show that the algorithm does not extend to more general potential games (that are not *exact* potential games). Also, and more interestingly in our case, it does not extend to other route update processes, such as synchronous schemes (see example 4 of their paper).

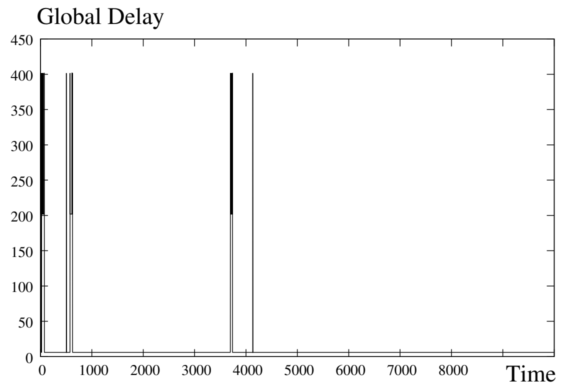

Figure 2: Convergence of the Gibbs sampling algorithm to the optimal association in the example.

IV. Network Implementation

The goal of this section is to show how this algorithm can be implemented in a fully distributed and asynchronous way.

First note that the algorithm assumes that repercussions can be computed by each node. Since the repercussions for flow u are sums of delays without flow u, they can be difficult to obtain. An option is to cut flow u as soon as it is chosen by the algorithm, measure the delays on each flow and then restart flow u on its new route. The main assumption here is that the time scale of the algorithm is slower than the one needed to to get a good estimate of the average delays of flows. Another needed assumption is that time is universal and clocks in all nodes are synchronized.This is essential to compute the delays from the time stamp on the packets.

Under these assumptions, Algorithm 2 can be transformed into a program implemented in real networks in a distributed way, where one part is executed by the nodes composing the network (computing the repercussion delays) and one part by every user, or more precisely by the source of each flow (run the Gibbs sampler).

Each packet from a given flow is stamped by the following list: its source, its route (sequence of nodes) and its release time. Packets from a given user follow their current route through the hops composing it. Messages are sent back to the source by each hop (using an arbitrary way back to the source). These very small backward messages are assumed to have a negligible impact on the network performance.

A. Network program

The role of the nodes composing the network is to compute the repercussion delays (3) and to transmit them to the users. This can be done in a distributed way. Since the algorithm is fully asynchronous, each link can compute the repercussion of the user u by the following infinitely repeated sequence of actions.

1) Measure the delays of all packets from all users.
2) Once user u stops using the node, measure the delays of the packets from all users without user u to get the repercussion prices.
3) Compute the repercussion delay given in Eq. (4)-(5).
4) Send the repercussion delay to the source of user u (this is often done by attaching this message to the next packet traveling to the source).

B. Users program

The users execute their part of the algorithm in a completely distributed and asynchronous way.

Here also, the algorithm is run as an infinite loop by each user in an asynchronous way. A method to do this is to introduce a random step size for each user with an exponential distribution with mean equal to RTT, as done in [15].

1) At the end of a time-step for user u,
2) The user stops sending packets on its current route.
3) It receives the messages from all the nodes on its current route. It sums the repercussion delays (note that the user has no way to know the real delay on the nodes). This is done over a time sliding window large enough so that the delay estimation is close enough to the expected delay.
4) The user runs the Gibbs sampler to choose a new route to transmit its packets
5) The user chooses the next time-step with an exponential distribution.

C. Measures based on loads

To avoid the complicated procedure used in the previous algorithm where users have to stop sending packets and nodes have to first measure the delays with and then without flow u, before sending the information to the source of u, it is possible to base the algorithm on delay estimations rather than on measurements.

Nodes measure the throughput for each flow u (number of packets arriving per second) instead of the delays of each packet of u. The average delay is estimated using formulas (4)-(5). Similarly, the repercussion of flow u is estimated without stopping flow u by using once more Formulas (4)-(5) in which flow u is removed.

This approach has several advantages compared with the true measures on delays. First it does not require that nodes have access to universal time (to measure delays). Second, the repercussions can be computed without stopping the flow and finally it does not require the time scales of the algorithm and of the delay estimation are of different orders of magnitude.

The main drawback of this new approach is that delays are only indirectly estimated rather than measured. Yet, since the number of route is limited for each flow, the choice of the best route does not require an accurate computation of the delay when the difference is large enough. In case both routes have similar delays, a precise computation may be needed to discriminate between both routes. However a bad choice will not deteriorate the global performance by much in that case.

V. Numerical results

In this section, we run several a large set of simulation to evaluate our algorithm, and compare it to the performance of AODV. We have used the Maple software to implement our algorithm.

Let us now describe the experimental road map and how the different parameters used in the simulations have been chosen (the units of the value are not specified).

Network Topology: The graph of the network is generated at random over 20 vertices: any two vertices are connected with a probability p and as long as the generated graph is not strongly connected, it is rejected . All the service rate of the arc are equal to 10.

Users: The number of users is 25 (defined as a pair of vertices). A user is defined as a flow from a source to a destination, and two different users may have the same pair source destination (but the application is not necessarily the same in both cases). The source and the destination of each flow are uniformly sampled among the set of nodes. Each user also has a level of priority and a rate. We only consider two levels of priority. A user is of priority 1 (with delay constraints) with probability 0.5 and has a flow rate λ, and of priority 2 (without delay constraints) otherwise, with a flow rate 5λ.

Delays: The delay is modeled by the mean delay of a packet in a M/M/1 queue with 2 classes of priorities, in a preemptive case (a packet in service will interrupt its service when a customer of a higher priority class arrives).

Denote by λ_1 and λ_2 the arrival rate of each priority flow, and assume that these flows are independent. Denote by μ the service rate of the node. Then the mean delays d_1 and d_2 for the packets of highest priority and lower priority are, respectively

$$d_1 = \frac{\lambda_1}{\mu - \lambda_1}, \tag{4}$$

$$\text{and } d_2 = \frac{1}{\lambda_2}\left(\frac{\lambda^2}{\mu-\lambda} - \frac{\lambda_1^2}{\mu-\lambda_1}\right), \tag{5}$$

where $\lambda = \lambda_1 + \lambda_2$. Indeed, the nonpreemptive hypothesis give the mean waiting time for priority class 1. By Little's law, we have $N_i = \lambda_i d_i$, where N_i is the mean number of packets of priority class i in the queue. As the arrival flows are independent, the global flow (independently of the priority class) is Poisson, then the mean sojourn time in the system is $d = \frac{\lambda}{\mu-\lambda}$. We can apply the Little law to the global process, *i.e.* $N = \lambda d$. Finally, $d_2 = \frac{1}{\lambda_2}N_2 = \frac{1}{\lambda_2}(N - N_1)$. This gives the result. Note that this computation can be extended easily for an arbitrary number of priority classes.

Set of paths: The set of paths for each user is chosen to be of cardinality less than 4. The first path is the one given by the actual AODV protocol, *i.e.* the one that minimizes the number of hops from the source to the destination. The

AODV protocol computes this path in a distributed fashion. The second path is given by AODV over a modified graph: the first arc of the previous path is removed[3]. The third path is obtained by removing the first arcs of the both previous paths. Similar for the fourth path.

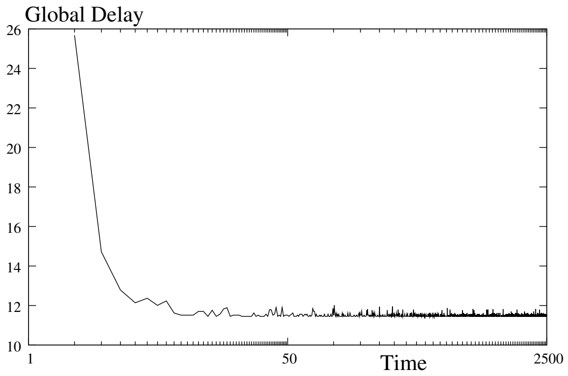

Figure 3: Convergence of a sample trajectory of the global delay, with the optimal Gibbs sampling algorithm.

Figure 3 displays a trajectory of the Gibbs sampling algorithm over one example, randomly chosen according to the procedure detailed above. Small jumps correspond to the occasional visits to weak associations, however, the global behavior is a fast convergence to the best association.

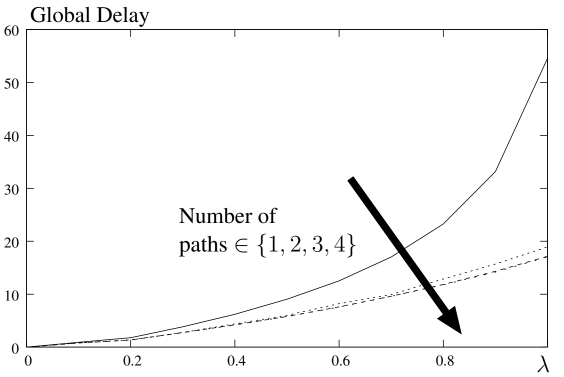

Figure 4: Global delay as a function of the total arrival rate in the network, for a fixed topology, with $1, 2, 3$ and 4 available paths.

Figure 4 displays the gain that can be obtained by adding one, two or three alternative routes to the AODV route. As the input rate of all flows grows (vertical axis), the gain increases when one route is added. While adding one route is very beneficial, the addition of a second route brings very little gain and a third alternative route is useless.

Figure 5 shows the number of steps needed for the Gibbs sampler to reach an association whose delay is less than 1% larger than the optimal association. Less than 50 iterations are sufficient in most cases, confirming the applicability of the technique in real ad-hoc systems. Furthermore, the distribution fits a Poisson distribution with parameter $\lambda = 18.76$ rather well.

[3]if no path connects the source to the destination in the modified graph, AODV does not return any additionnal path

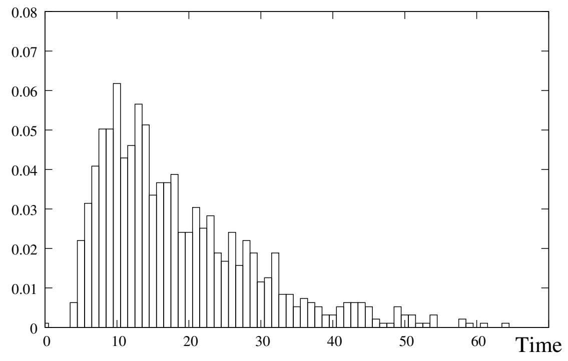

Figure 5: Distribution of the maximal number of iteration per user to reach an association with a delay less than 1% larger than the optimal one.

VI. Conclusion

This paper shows that performance of Ad-hoc networks can get substantial improvements by replacing AODV routes for each flow by a choice between a small number of alternatives and pick the best among them. This choice can be based on a Gibbs sampling algorithm run by every flow independently and is guaranteed to convergence to a social optimal configuration.

References

[1] K. Ghada, J. Li, and Y. Ji, "Cross-layer approach for energy efficient routing in wanets," in *IEEE MASS*, 2009.
[2] I. D. Chakeres and E. M. Belding-Royer, "Aodv routing protocol implementation design," in *Proceedings of the 24th International Conference on Distributed Computing Systems*, 2004.
[3] H. Kaaren, A. Ahtiainen, L. Laitinen, S. Naghian, and V. Niemi, *UMTS networks: architecture, mobility, and services*. Wiley, 2005.
[4] D. Monderer and L. S. Shapley, "Potential games," *Games and Economic Behavior*, vol. 14, pp. 124–143, 1996.
[5] B. Vocking, "Congestion games: Optimization in competition," in *Proceedings of the 2nd Algorithms and Complexity in Durham Workshop*, 2006.
[6] B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, D. Papagiannaki, and C. Diot, "Measurement-based self organization of interfering 802.11 wireless access networks," in *IEEE Infocom*, 2007.
[7] B. Hajek, "Cooling schedules for optimal annealing," *MATHEMATICS OF OPERATIONS RESEARCH*, vol. 13, pp. 311–329, 1988.
[8] J. Galtier and A. Laugier, "Flow on data network and a positive semidefinite representable delay function," *Journal of Interconnection Networks*, vol. 8, pp. 29–43, 2007.
[9] R. W. Rosenthal, "A class of games possessing pure-strategy Nash equilibria," *Int. J. Game Theory*, vol. 2, pp. 65–67, 1973.
[10] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," in *Proc. of STACS*, 1998.
[11] T. Roughgarden and E. Tardos, "Bounding the inefficiency of equilibria in nonatomic congestion games," *GAMES AND ECONOMIC BEHAVIOR*, vol. 47, pp. 389–403, 2004.
[12] R. Cominetti, J. R. Correa, and N. E. Stier-Moses, "The impact of oligopolistic competition in networks," *OPERATIONS RESEARCH*, vol. 57, pp. 1421–1437, 2009.
[13] P. Bremaud, *Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, 1999.
[14] C. Alos-Ferrer and N. Netzer, "The logit-response dynamics," Thurgauer Wirtschaftsinstitut, Universitat Konstanz, Tech. Rep., 2008.
[15] C. S. Chen and F. Baccelli, "Self-optimization in mobile cellular networks: power control and user association," *IEEE International Conference on Communications*, pp. 1–6, 2010.