Cooperation Enforcement and Learning for Optimizing Packet Forwarding in Autonomous Wireless Networks

Charles Pandana*, Zhu Han+, and K. J. Ray Liu
* Arraycomm, San Jose, CA 95131
+ Department of Electrical and Computer Engineering, Boise State University, Boise, ID 83706
Department of Electrical and Computer Engineering, and Institute of Systems Research
University of Maryland, College Park, MD 20742

***Abstract*— In wireless ad hoc networks, autonomous nodes are reluctant to forward others' packets because of the nodes' limited energy. However, such selfishness and noncooperation deteriorate both the system efficiency and nodes' performances. Moreover, the distributed nodes with only local information may not know the cooperation point, even if they are willing to cooperate. Hence, it is crucial to design a distributed mechanism for enforcing and learning the cooperation among the greedy nodes in packet forwarding. In this paper, we propose a self-learning repeated-game framework to overcome the problem and achieve the design goal. We employ self-transmission efficiency as the utility function of individual autonomous node. The self transmission efficiency is defined as the ratio of the power for self packet transmission over the total power for self packet transmission and packet forwarding. Then, we propose a framework to search for good cooperation points and maintain the cooperation among selfish nodes. The framework has two steps: First, an adaptive repeated game scheme is designed to ensure the cooperation among nodes for the current cooperative packet forwarding probabilities. Second, self-learning algorithms are employed to find the better cooperation probabilities that are feasible and benefit all nodes. We propose three learning schemes for different information structures, namely, learning with perfect observability, learning through flooding, and learning through utility prediction. Starting from noncooperation, the above two steps are employed iteratively, so that better cooperating points can be achieved and maintained in each iteration. From the simulations, the proposed framework is able to enforce cooperation among distributed selfish nodes and the proposed learning schemes achieve 70% to 98% performance efficiency compared to that of the optimal solution.**

I. Introduction

Some wireless networks such as ad-hoc networks consist of autonomous nodes without centralized control. In such autonomous networks, the nodes may not be willing to fully cooperate and accomplish the network task. Specifically for the packet forwarding problem, forwarding the others' packets consumes the node's limited battery resource. Therefore, it may not be of the node's best interest to forward others' arriving packets. However, rejection of forwarding others' packets non-cooperatively will severely affect the network functionality and impair the nodes' own benefits. Hence, it is crucial to design a mechanism to enforce cooperation among greedy nodes. In addition, the randomly located nodes with local information may not know how to cooperate, even if they are willing to cooperate.

This work is partially supported by MURI AFOSR F496200210217.

The packet forwarding problem in ad hoc networks has been extensively studied in the literature. The fact that nodes act selfishly to optimize their own performances has motivated many researchers to apply the game theory [1], [2] in solving this problem. Broadly speaking, the approaches used in encouraging the packet forwarding task can be categorized into two methods. The first type of methods makes use of virtual payment. Virtual currency, pricing, and credit based method [3], [4] fall into this first type. The second type of approaches is related to personal and community enforcement to maintain the long-term relationship among nodes. Cooperation is sustained because defection against one node causes personal retaliation or sanction by others. This second approach includes the following works. Marti et al. [5] propose mechanism called *watchdog* and *pathrater* to identify the misbehaving nodes and deflect the traffic around them. Buchegger et al. [6] define protocols based on reputation system. Altman et al. [7] consider a punishment policy to show cooperation among participating nodes. In [8], Han et al. propose learning repeated game approaches to enforce cooperation and obtain better cooperation solutions. Some other works using game theory in solving communication problems can be found in [9], [10], and [11].

Since in some wireless networks, it is difficult to implement the virtual payment system because of the practical implementation challenges such as enormous signaling. In this paper, we concentrate on the second type of approaches and design a mechanism such that cooperation can be enforced in a distributed way. In addition, unlike the previous works which assume the nodes know the cooperation points or other nodes' behaviors, we argue that randomly deployed nodes with local information may not know how to cooperate even if they are willing to do so. Motivated by these facts, we propose a self-learning repeated-game framework for cooperation enforcing and learning.

We define the self-transmission as the transmission of a user's own packets. We quantify the node's utility as its self-transmission efficiency, which is defined as the ratio of the power for successful self transmission over the total power used for self transmission and packet forwarding. The goal of the node is to maximize the long-term average efficiency. Using this utility function, a distributed self-learning repeated-game framework is proposed to ensure cooperation among autonomous nodes. The framework consists of two steps: First, the repeated game enforces cooperation in packet forwarding.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2008**	2. REPORT TYPE	3. DATES COVERED **00-00-2008 to 00-00-2008**
4. TITLE AND SUBTITLE **Cooperation Enforcement and Learning for Optimizing Packet Forwarding in Autonomous Wireless Networks**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Maryland,Department of Electrical Engineering and Computer Engineering,Institute for Systems Research,College Park,MD,20742**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S) 11. SPONSOR/MONITOR’S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		

14. ABSTRACT

In wireless ad hoc networks, autonomous nodes are reluctant to forward others’ packets because of the nodes’ limited energy. However, such selfishness and noncooperation deteriorate both the system efficiency and nodes’ performances. Moreover, the distributed nodes with only local information may not know the cooperation point, even if they are willing to cooperate. Hence, it is crucial to design a distributed mechanism for enforcing and learning the cooperation among the greedy nodes in packet forwarding. In this paper, we propose a self-learning repeated-game framework to overcome the problem and achieve the design goal. We employ self-transmission efficiency as the utility function of individual autonomous node. The self transmission efficiency is defined as the ratio of the power for self packet transmission over the total power for self packet transmission and packet forwarding. Then, we propose a framework to search for good cooperation points and maintain the cooperation among selfish nodes. The framework has two steps: First, an adaptive repeated game scheme is designed to ensure the cooperation among nodes for the current cooperative packet forwarding probabilities. Second, self-learning algorithms are employed to find the better cooperation probabilities that are feasible and benefit all nodes. We propose three learning schemes for different information structures, namely, learning with perfect observability learning through flooding, and learning through utility prediction. Starting from noncooperation, the above two steps are employed iteratively, so that better cooperating points can be achieved and maintained in each iteration. From the simulations, the proposed framework is able to enforce cooperation among distributed selfish nodes and the proposed learning schemes achieve 70% to 98% performance efficiency compared to that of the optimal solution.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**13**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

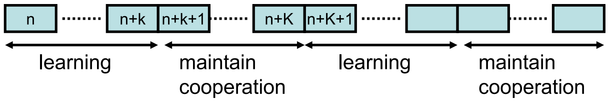


Fig. 1. Illustration of time-slotted transmission to two alternative stages

This first step ensures that any cooperation equilibrium that is more efficient than the Nash Equilibrium (NE) of the one stage game can be sustained. The repeated game allows nodes to consider the history of actions/reactions of their opponents in making the decision. The cooperation can be enforced/sustained using the repeated game, since any deviation causes the punishment from other nodes in the future. The second step utilizes the learning algorithm to achieve the desired efficient cooperation equilibrium. We propose three learning algorithms for different information structures, namely, learning with perfect observability, learning through flooding, and learning through utility prediction. Starting from the non-cooperation point, the two proposed steps are applied iteratively. A better cooperation is discovered and maintained in each iteration, until no more efficient cooperation point can be achieved. From the simulation results, our proposed framework is able to enforce cooperation among selfish nodes. Moreover, compared to the optimal solution obtained by a centralized system with global information, our proposed learning algorithms achieve similar performances in the symmetric network. Depending on learning algorithms and the information structures, our proposed schemes achieve near-optimal solution in the random network.

This paper is organized as follows: In Section II, we give the system model and explain the design challenge. In Section III, we propose and analyze the repeated-game framework for packet forwarding under different information structures. In Section IV, we construct self-learning algorithms corresponding to different information structures in details. In Section V, we evaluate the performances of our proposed scheme using extensive simulations. Finally, the conclusions are drawn in Section VI.

II. System Model and Design Challenge

We consider a network with N nodes. Each node is battery-powered and has transmit power constraint. This implies that only nodes within the transmission range are neighbors. The packet delivery typically requires more than one hop. In each hop, we assume transmission occurs in a time-slotted manner as illustrated in Figure 1. The source, the relays (intermediate nodes), and the destination constitute an active route. We assume an end-to-end mechanism that enables a source node to know if the packet is delivered successfully. The source node can observe whether there is a packet drop in one particular active path. However, the source node may not know where the packet is dropped. Finally, we assume that routing decision has already been done before optimizing the packet forwarding probabilities[1].

[1]We note that it is always possible for nodes to do manipulation in the routing layer. However, it is beyond the scope of this paper. For more information, please refer to [16]

Let's denote the set of sources and destinations as $\{S_i, D_i\}$, for $i = 1, 2, \cdots, M$, where M represents the number of source-destination pairs that are active in the network. Suppose the shortest path for each source-destination pair has been discovered. Let's denote the route/path as $R_i = (S_i, f^1_{R_i}, f^2_{R_i}, \cdots, f^n_{R_i}, D_i)$, where S_i denotes the source node, D_i denotes the destination node, and $\{f^1_{R_i}, f^2_{R_i}, \cdots, f^n_{R_i}\}$ is the set of intermediate/relay nodes, thus, there are $n+1$ hops from source node to the destination node. Let $V = \{R_i : i = 1, \cdots, M\}$ be the set of routes corresponding to all source-destination pairs. Let's denote further the set of routes where node j is the source as $V^s_j = \{R_i : S(R_i) = j, i = 1 \ldots M\}$, where $S(R_i)$ represents the source of route R_i. The power expended in node i for transmitting its own packet is

$$P^{(i)}_s = \sum_{r \in V^s_i} \mu_{S(r)} \cdot K \cdot d(S(r), n(S(r), r))^\gamma, \qquad (1)$$

where $\mu_{S(r)}$ is the transmission rate of source node $S(r)$, K is the transmission constant, $d(i, j)$ is the distance between node i and node j, $n(i, r)$ denotes the neighbor of node i on route r, and γ is the transmission path-loss coefficient. For the link from node i to its next hop $n(i, r)$ on route r, $K \cdot d(i, n(i, r))^\gamma$ describes the reliable successful transmission power per bit transmission. We note that equation (1) can also be interpreted as the average signal power required for successful transmission of certain rate $\mu_{S(r)}$. This implies that the transmission failure due to the channel fading has been taken into account by the transmission constant K.

Let α_i for $i = 1, \cdots, N$ be the packet forwarding probability for node i. Here, we use the same packet forwarding probability for every source-destination pairs because of the following reasons. First, based on the greedy assumption of the nodes, there is no reason for one particular node to forward some packets on some routes and reject forwarding other packets on other routes. Second, the use of different packet forwarding probability on different routes will only complicate the deviation detection of a node and it will not change the optimization framework proposed in this paper. So in our first step to analyze the problem, we assume the same forwarding probability on every route. In the future work, we are also exploring the case where the nodes use different packet forwarding probability for different routes.

Clearly, probability of successful transmission from node i to its destination depends on the forwarding probabilities employed in the intermediate nodes and it can be represented as

$$P^i_{Tx,r} = \prod_{j \in (r \setminus \{S(r)=i, D(r)\})} \alpha_j, \qquad (2)$$

where $D(r)$ is the destination of route i and $(r \setminus \{S(r) = i, D(r)\})$ is the set of nodes on route r excluding the source and destination. Let's define the *good power* consumed in transmission node i, $P^{(i)}_{s,good}$ as the product of the power used for transmitting node i's own packet and the probability of successful transmission from node i to its destination,

$$P^{(i)}_{s,good} = \sum_{r \in V^s_i} \mu_{S(r)} \cdot K \cdot d(S(r), n(S(r), r))^\gamma P^i_{Tx,r}. \qquad (3)$$

Moreover, let the set of routes where node j is the forwarding node be W_j. The power used to forward others' packets is given by

$$P_f^{(i)} = \alpha_i \cdot K \cdot \sum_{r \in W_i} d(i, n(i,r))^{\gamma} \mu_{S(r)} P^i_{F,r}, \qquad (4)$$

where $P^i_{F,r}$ is the probability that node i receives the packet to forward in route r, and $\sum_{r\in W_i} \mu_{S(r)}\ P^i_{F,r}$ is the total rate that node i receives for packet forwarding. The probability that node i receives the forward packet in route r is represented as

$$P^i_{F,r} = \prod_{j\in\{f_r^1, f_r^2, \cdots, f_r^{m-1}\}} \alpha_j, \qquad (5)$$

where $r = \{S(r), f_r^1, \cdots, f_r^{m-1}, f_r^m = i, \cdots, f_r^n, D(r)\}$ is the $n+1$ hops route from source $S(r)$ to destination $D(r)$, and the m^{th} forwarding node f_r^m is node i. $P^i_{F,r}$ depends on the packet forwarding probabilities of the nodes on the route r before node i.

We refer to the task of transmitting the node own information as self-transmission and the task of relaying others' packets as packet forwarding. We focus on maximizing the *self-transmission efficiency*, which is defined as the ratio of successful self-transmission power (good power) over the total power used for self-transmission and packet forwarding. Therefore, the stage utility function for node i can be represented as

$$U^{(i)}(\alpha_i, \alpha_{-i}) = \frac{P^{(i)}_{s,good}}{P_s^{(i)} + P_f^{(i)}}. \qquad (6)$$

where α_i is node i's packet forwarding probability, $\alpha_{-i} = (\alpha_1, \cdots, \alpha_{i-1}, \alpha_{i+1}, \cdots, \alpha_N)^T$ are the other nodes' forwarding probability. Putting (1), (3) and (4) into (6), we obtain (7). Since the power for successful self-transmission depends on the packet forwarding used by other nodes, the self-transmission efficiency captures the trade-off between the power used for packet transmission of its own information and packet forwarding for the other nodes.

The problem in packet forwarding arises because the autonomous nodes such as in ad-hoc networks have their own authorities to decide whether to forward the incoming packets. Under this scenario, it is very natural to assume that each node selfishly optimizes its own utility function. In parallel to (7), node i selects α_i in order to maximize the transmission efficiency $U^{(i)}(\alpha_i, \alpha_{-i})$. This implies that node i will selfishly minimize $P_f^{(i)}$, the portion of energy used to forward others' packets. In the game theory literature [1], [2], Nash Equilibrium (NE) is a well-known concept, which states that in the equilibrium every node selects the best response strategy to the other nodes' strategies. The formal definition of NE is given as follow

Definition 1: Define feasible range Ω as $[0,1]$. Nash Equilibrium $[\alpha_1^*, \cdots, \alpha_N^*]^T$ is defined as:

$$U^{(i)}(\alpha_i^*, \alpha_{-i}^*) \geq U^{(i)}(\alpha_i, \alpha_{-i}^*), \forall i,\ \forall \alpha_i \in \Omega, \qquad (8)$$

i.e., given that all nodes play NE, no node can improve its utility by unilaterally changing its own packet forward probability. Here $\alpha_{-i}^* = (\alpha_1^*, \cdots, \alpha_{i-1}^*, \alpha_{i+1}^*, \cdots, \alpha_N^*)^T$.

Unfortunately, the NE for the packet forwarding game described in (7) is $\alpha_i^* = 0,\ \forall i$. This can be verified by finding the forwarding probability $\alpha_i \in [0,1]$ such that $U^{(i)}$ is unilaterally maximized. To maximize the transmission efficiency of node i, the node can only make the forwarding energy $P_f^{(i)}$ as small as possible. This is equivalent to setting α_i as small as possible, since the successful probability of its own packet transmission in (2) depends only on the other nodes' willingness to forward the packets. By greedily dropping its packet forwarding probability, node i reduces its total transmission power used for forwarding others' packets, therefore, increases its instantaneous efficiency. However, if all nodes play the same strategy, this causes zero efficiency in all nodes, i.e., $U^{(i)}(\alpha_1^* \ldots \alpha_N^*) = 0,\ \forall i$. As the result, the network breaks down. Hence, playing NE is inefficient not only from the network point of view but also for the individual's own benefit. It is very important to emphasize that the inefficiency of NE is independent to the utility function in (7). This inefficiency is merely the result of greedy optimization unilaterally done by each of the nodes. In the next two sections, we propose a self-learning repeated-game framework and show how cooperation can be enforced using our proposed scheme.

III. Repeated-Game Framework and Punishment Analysis

As demonstrated in Section II, the packet forwarding game has $\alpha_i^* = 0,\ \forall i$ as its unique Nash equilibrium if the game is only played once. This implies that all nodes in the network won't be cooperating in forwarding the packets. In practice, nodes typically participate in the packet forwarding game for a certain duration of time, and this is *more suitably* modelled as a repeated game (a game that is played in multiple times). If the game never ends, it is called infinite repeated game which we will use in this paper. In fact, the repeated game may not be necessarily infinite. The important point is that the nodes/players do not know when the game ends. In this sense, the properties of the infinitely repeated game can still be valid. In this paper, we employ the normalized average discounted utility with discounting factor δ given by:

$$\bar{U}_\infty^{(i)} = \lim_{t'\to\infty} \bar{U}_{t'}^{(i)} = (1-\delta) \sum_{t=1}^{\infty} \delta^{(t-1)} U^{(i)}(\vec{\alpha}(t)), \qquad (9)$$

where $\vec{\alpha}(t) = (\alpha_1, \ldots, \alpha_N)^T$, $U^{(i)}(\vec{\alpha}(t))$ is the utility of node i at each stage game (7) played at time t, and $\bar{U}_{t'}^{(i)}$ is the normalized average discounted utility from time 1 to time t'. Unlike the one-time game, the repeated game allows a strategy to be contingent on the past moves and results in the reputation and retribution effects, so that cooperation can be sustained [2], [13], [14]. We also note that the utilities in (7) and (9) are indeed heterogeneous in the sense that they carry the information about the channel, routing, and node behaviors. In other words, the utility functions in (7) and (9) reflect different energy consumption according to different distance, rate, and route between nodes.

A. Design of Punishment Scheme under Perfect Observability

In this subsection, we analyze a class of punishment policy under the assumption of perfect observability. Perfect observ-

$$U^{(i)} = \frac{\sum_{r\in V_i^s} \mu_{S(r)} d(S(r), n(S(r), r))^\gamma \prod_{j\in (r\setminus\{S(r)=i, D(r)\})} \alpha_j}{\sum_{r\in V_i^s} \mu_{S(r)} d(S(r), n(S(r), r))^\gamma + \alpha_i \sum_{r\in W_i} d(i, n(i, r))^\gamma \mu_{S(r)} \prod_{j\in\{f_r^1,\cdots,f_r^{m-1}\}} \alpha_j}. \tag{7}$$

ability means that each node is able to observe actions taken by other nodes along the history of the game. This implies that node knows which node drops the packet and is aware of the identity of other nodes. This condition allows every node to detect any defection of other nodes and it also allows nodes to know if any node does not follow the game rule. The perfect observability is the ideal case and serves as the performance upper bound. In the next subsection, this assumption is relaxed to a more practical situation, where an individual node only has limited local information.

Let's denote the NE in one stage forwarding game as $\vec{\alpha}^* = (\alpha_1^*, \cdots, \alpha_N^*)^T$, and the corresponding utility functions as $(v_1^*, \cdots, v_N^*)^T = (U^{(1)}(\vec{\alpha}^*), \cdots, U^{(N)}(\vec{\alpha}^*))^T$. We also denote

$$\begin{aligned} \mathbf{U} &= \{(v_1,\cdots,v_N)|\ \exists \vec{\alpha} \in \Omega^N \qquad (10)\\ &\quad \text{s.t.} \quad (v_1,\cdots,v_N) = (U^{(1)}(\vec{\alpha}),\cdots,U^{(N)}(\vec{\alpha}))\},\\ \mathbf{V} &= \texttt{convex hull of } \mathbf{U}, \qquad (11)\\ \mathbf{V}^\dagger &= \{(v_1,\cdots,v_N) \in \mathbf{V}|\ v_i > v_i^*,\ \forall i\}. \qquad (12)\end{aligned}$$

We note that $\mathbf{V}$ consists of all feasible utilities, and $\mathbf{V}^\dagger$ consists of feasible utilities that Pareto-dominate the one stage NE, this set is also known as the individually rational utility set [1], [2]. The Pareto-dominant utilities denote all utilities that are strictly better than the one stage NE. From the game theory literature [2], [13], [14], the existence of equilibria that Pareto-dominate the one stage NE is given by the Folk theorem [14].

Theorem 1 (Folk Theorem [14]): Assume that the dimensionality of $\mathbf{V}^\dagger$ equals to N. Then, for any $(v_1, \cdots, v_N)$ in $\mathbf{V}^\dagger$, there exists $\underline{\delta} \in (0, 1)$ such that for all $\delta \in (\underline{\delta}, 1)$, there exists an equilibrium of the infinitely repeated game with discounted factor δ in which player i's average utility is v_i.

Before we give the application of Folk theorem in the packet forwarding game, it is useful to recall the notion of *dependency graph*. Given the routing algorithm and the source-destination pairs, the dependency graph is the directed graph that is constructed as follows. The number of nodes in the dependency graph is the same as the number of nodes in the network. When node i sends packets to node j via nodes $f^1, \cdots, f^n$, then there exist directed edges from node i to nodes $f^1, \cdots, f^n$. The resulting dependency graph is a directed graph, which describes the node dependency in performing the packet forwarding task. Let's define $deg_{in}(i)$ and $deg_{out}(i)$ as the number of edges going into node i and coming out from node i, respectively. Obviously, $deg_{in}(i)$ indicates of the number of nodes whose packets are forwarded by node i and $deg_{out}(i)$ is the number of nodes that help forward node i's packets. Using the notation of the corresponding dependency graph, the application of Folk theorem in packet forwarding game is stated as follow:

Theorem 2: (Existence of Pareto-dominant forwarding equilibria under perfect observability)

Under the following conditions

1. the game is perfectly observable;
2. the corresponding dependency graph satisfies the condition
$$deg_{out}(i) > 0,\ \forall i; \tag{13}$$
3. $\mathbf{V}^\dagger$ has full dimensionality ($\mathbf{V}^\dagger$ has dimensionality of N). We note that $\mathbf{V}^\dagger$ has dimensionality of N implies that the space formed by all points in $\mathbf{V}^\dagger$ has the dimensionality of N.

Then, for any $(v_1, \cdots, v_n) \in \mathbf{V}^\dagger$, there exists $\underline{\delta} \in (0, 1)$, such that for all $\delta \in (\underline{\delta}, 1)$, there exists an equilibrium of the infinitely repeated game with node i's average utility v_i.

Proof: Let $\vec{\alpha} = (\alpha_1, \cdots, \alpha_N)^T$ be the joint strategy results in $(U^{(1)}(\vec{\alpha}), \cdots, U^{(N)}(\vec{\alpha}))$. The full dimensionality condition ensures the set $\big(U^{(1)}(\vec{\alpha}), \cdots, U^{(j-1)}(\vec{\alpha}), U^{(j)}(\vec{\alpha}) - \varepsilon, U^{(j+1)}(\vec{\alpha}), \cdots,\ U^{(N)}(\vec{\alpha})\big)$ for any $\varepsilon > 0$, is in $\mathbf{V}^\dagger$. Let node i's maximum utility be $\overline{v}_i = \max_{\vec{\alpha}} U^{(i)}(\vec{\alpha}),\ \forall i$. This maximum utility is obtained when all nodes try to maximize node i's utility. Let the cooperating utility be $v_i = U^{(i)}(\vec{\alpha}) \in \mathbf{V}^\dagger, \forall i$. The cooperating utilities are obtained when all nodes play the agreed packet forwarding probabilities. Let the maximum utility node i can get when it is punished be $\underline{v}_i = \max_{\alpha_i} \min_{\alpha_{-i}} U^{(i)}(\vec{\alpha})$. Let's denote node j's utility when punishing node i as w_j^i. We note that from (7), the max-min utility $\underline{v}_i$ coincides with the one stage NE. If there exist ϵ and the punishment period for node i, T_i, such that

$$\frac{\overline{v}_i}{U^{(i)} - \epsilon} < (1 + T_i), \tag{14}$$

then the following rules ensure any individually rational utilities can be enforced.

1) *Condition I:* All nodes play cooperation strategies if there is no deviation in the last stages. After any deviations go to Condition II (Suppose node j deviates).
2) *Condition II:* Nodes that can punish the deviating node (node j) play the punishing strategies for the punishment period. The rest of the nodes keep playing cooperating strategies. If there is any deviation in Condition II, restart Condition II and punish the deviating node. If any punishing node does not play punishment in the punishment period, the other nodes will punish that particular node during the punishment period. Otherwise, after the end of the punishment period, go to Condition III.
3) *Condition III:* Play strategy that results in utility $\big(U^{(1)}, \cdots, U^{(j-1)}, U^{(j)} - \varepsilon, U^{(j+1)}, \cdots,\ U^{(N)}\big)$. If there is any deviation in Condition III, start Condition II and punish the deviating node.

First, the cooperating strategy is the strategy that all nodes agree upon. In contrast, the punishing node i strategy, is the strategy that results in max-min utility in node i, $\underline{v}_i = \max_{\alpha_i} \min_{\alpha_{-i}} U^{(i)}(\vec{\alpha})$. In the sequel, we show that under the proposition's assumptions:

- the average efficiency gained by the deviating node is smaller than the cooperating efficiency,
- the average efficiency gained by the punishing node that does not play the punishment strategy in the punishment stage is worse than the efficiency gained by that node when it conforms to the punishing strategy.

If node j deviates in Condition I and then conforms, it receives at most $\overline{v}_j$ when it deviates, $\underline{v}_j$ for T_j periods when it is punished, and $(U^{(j)} - \varepsilon)$ after it conforms to the cooperative strategy. The average discounted deviation utility can be expressed as:

$$\hat{U}_{\infty}^{(j)} = \overline{v}_j + \frac{\delta(1-\delta^{T_j})}{1-\delta}\underline{v}_j + \frac{\delta^{T_j+1}}{1-\delta}(U^{(j)} - \varepsilon). \tag{15}$$

Since if the node conforms throughout the game, it has the average discounted utility of $\frac{1}{1-\delta}U^{(j)}$. So the gain of deviation is given by:

$$\Delta U^{(j)} = \hat{U}_{\infty}^{(j)} - \frac{1}{1-\delta}U^{(j)}$$
$$< \overline{v}_j + \frac{\delta(1-\delta^{T_j})}{1-\delta}\underline{v}_j - \frac{1-\delta^{T_j+1}}{1-\delta}(U^{(j)} - \varepsilon). \tag{16}$$

We note that $\underline{v}_j$ coincides with the one stage NE, which is $\underline{v}_j = 0, \forall j$. As $\delta \to 1$, $\frac{1-\delta^{T_j+1}}{1-\delta}$ tends to $1 + T_j$. Under the condition of (14), the deviation gain in (16) will be strictly less than zero. This indicates that the average cooperating efficiency is *strictly* larger than the deviation efficiency. Hence, any rational node will not deviate from the cooperation point.

If the punished node still deviates in the punishment period, the punishment period (Condition II) restarts and the punishment duration experienced by the punished node is lengthened. As the result, deviation in the punishment period postpones the punished node from receiving the strictly better utility $(U^{(j)} - \varepsilon)$ in Condition III. Hence, it is better not to deviate in the punishment stage.

On the other hand, if punishing node i does not play the punishing strategy during the punishment of node j, node i receives at most

$$\hat{U}_{\infty}^{(i)} = \overline{v}_i + \frac{\delta(1-\delta^{T})}{1-\delta}\underline{v}_i + \frac{\delta^{T+1}}{1-\delta}(U^{(i)} - \varepsilon). \tag{17}$$

However, if node i conforms with the punishment strategy, it will receive at least

$$\tilde{U}_{\infty}^{(i)} = \frac{(1-\delta^{T})}{1-\delta}w_i^j + \frac{\delta^{T+1}}{1-\delta}U^{(i)}. \tag{18}$$

Here w_i^j is the utility of node i to punish node j. Therefore, node i's reward for carrying out the punishment is (18) minus (17),

$$\tilde{U}_{\infty}^{(i)} - \hat{U}_{\infty}^{(i)} = \frac{(1-\delta^{T})}{1-\delta}(w_i^j - \delta\underline{v}_i) - \overline{v}_i + \frac{\delta^{T+1}\varepsilon}{1-\delta}. \tag{19}$$

Using $\underline{v}_i = 0, \forall i$ and let $\delta \to 1$, the expression (19) is equivalent to

$$\tilde{U}_{\infty}^{(i)} - \hat{U}_{\infty}^{(i)} = T \cdot w_i^j - \overline{v}_i + \frac{\varepsilon}{1-\delta}. \tag{20}$$

By selecting δ close to one, this expression can be always larger than zero. As the result, the punishing node always conforms to the punishment strategy in the punishment stage.

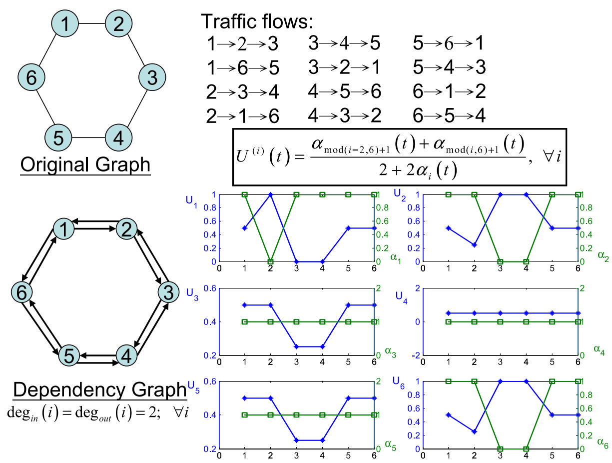


Fig. 2. Example of the punishment scheme under perfect observability

The same argument of no node deviating in Condition I can be used to show that no nodes deviates in Condition III. Therefore, we conclude that deviations in all Conditions are not profitable. ∎

The proof above is based on two conditions: First, the proof assumes that there always exist nodes that can punish the deviating nodes, this is guaranteed by the assumption $deg_{out}(i) > 0$ in the corresponding dependency graph. Secondly, nodes are able to identify which node is defecting and which node does not carry out the punishment. This is guaranteed by the perfect observability assumption. The strategy of punishing those who misbehave and those who do not punish the misbehaving nodes can be an effective strategy to cope with the collusion attack.

Now let's consider the following example to understand the punishment behavior. We assume $\mu_{S(r)} = 1$, $K = 1$, and $d(i,j) = 1$. The resulting utilities are shown in Figure 2. Each node has the one stage utility as:

$$U^{(i)} = \frac{\alpha_{\mathrm{mod}(i-2,6)+1} + \alpha_{\mathrm{mod}(i,6)+1}}{2 + 2\alpha_i}. \tag{21}$$

By selecting the discounted factor, $\delta = 0.9$ and $T = 2$ appropriately, all nodes are better-off when they are cooperating in packet forwarding by setting $\alpha_i = 1, \forall i$. If all nodes conform to the cooperative strategies, the 6-stage normalized average discounted utilities defined in (9) are given by $\bar{U}_6^{(i)} = 0.2343, \ \forall i$. In Figure 2, we plot the utility functions and forwarding probabilities of all nodes. The x-axis of the plot denotes the round of game, the left y-axis denotes the value of node's utility, and the right y-axis denotes the value of forwarding probability. The forwarding probability is denoted by the squared plot and the utility function is denoted by the plot with stars.

In Figure 2, we show that node 1 is deviating in the second round of the game by setting its forwarding probability to zero. At this time, node 1's utility changes from 0.5 to 1 as seen in the figure. As the consequence, node 2 and node 6 are punishing node 1 at the following $T = 2$ stages by setting their forwarding probabilities to zeros. In the third round of the game, node 1 has to return to cooperation. Otherwise, the punishment from others restarts and consequently the average discounted utility will be further lowered. After the punishment, all nodes come back to the cooperative forwarding probabilities (as shown in the figure). The resulting 6-stage normalized average utilities are as follows $\bar{U}_6^{(1)} = 0.2023$, $\bar{U}_6^{(2)} = \bar{U}_6^{(6)} = 0.2887$, $\bar{U}_6^{(3)} = \bar{U}_6^{(5)} = 0.1958$, and $\bar{U}_6^{(4)} = 0.2343$. So node 1 has less utility by deviation than by cooperation. Moreover, if both node 2 and node 6 fail to punish node 1, they will be punished by other nodes during the following T periods of game. The resulting normalized average utilities are $\bar{U}_6^{(1)} = 0.3485$, $\bar{U}_6^{(2)} = \bar{U}_6^{(6)} = 0.1425$, $\bar{U}_6^{(3)} = \bar{U}_6^{(5)} = 0.3035$, and $\bar{U}_6^{(4)} = 0.165$. Therefore, node 2 and node 6 will carry out the punishment, since otherwise they will in turn be punished and have less utility. The same argument can be used to prevent nodes deviating from the punishment strategy. We note that in this example the corresponding dependency graph has $deg_{in}(i) = deg_{out}(i) = 2, \forall i$. Therefore, there are always punishing nodes available whenever any node deviates.

Finally, we discuss the discounting factor δ which represents the *importance of the future*. In the case where the discounting factor is small, the future is less important. This will cause the pathological situation where the instantaneous deviation gain of the defecting node exceeds any future punishment by the other nodes. Therefore, it is better-off for the node to deviate rather than to cooperate and it becomes very hard (if not impossible) to encourage all nodes to cooperate in this scenario. We also note that the selfish nodes are better-off to choose the δ approaching to one. Since if the node chooses δ that closes to zero, this implies that the future is not important to the node, the node will definitely ask other nodes for transmitting his own packet at very beginning of the game and stop forwarding others' packets afterward. This will invoke punishment from its neighboring nodes by not forwarding that particular node's packets. This implies that that node will automatically be excluded from the network. Therefore, it is better-off for nodes in the network to choose δ approaching to one.

B. Design of Punishment Scheme under Imperfect Local Observability

We have shown that under the perfect observability assumption, the packet forwarding game along with the punishment scheme can achieve any Pareto-dominant efficiency. However, the perfect observability may be difficult to implement in ad-hoc networks, due to the enormous overheads and signaling. Therefore, we try to relax the condition of the perfect observability in this subsection. There are many difficulties in removing the perfect observability assumption. Suppose each node observes only its own history of stage utility function. In this situation, the node knows nothing about what has been going on in the rest of the network. The node only knows the deviation of nodes on which it relies on to do packet forwarding. And it cannot detect the deviation in the other part of the network, even though it can be the one that can punish the deviating node. Therefore, it is impossible to implement the Folk Theorem in this information limited situation. Moreover, nodes may not know if the system is in punishment stage or not. As soon as one of the nodes sees the deviation, it starts the punishment period. This will quickly start another punishment stage by other nodes, since the nodes cannot differentiate if the change in stage efficiency is caused by the punishment stage or the deviating node. As the result, the defection spreads like an epidemic and cooperation in the whole network breaks down. This is known as the *contagious equilibrium* [13]. Indeed, the only equilibrium in this situation is the one stage NE.

The main reason of the contagious equilibrium is that all nodes have the *inconsistent* beliefs about the state of the system, they do not know whether the system is currently in the punishment stage, the deviation state, or the end of punishment stage. Therefore, any mistake in invoking the punishment stage can cause the contagious equilibrium. The lack of the consistent knowledge of the system state can be mitigated using communications between nodes. Suppose each node observes only a subset of the other nodes' behaviors. The communication is introduced by assuming that each node makes a public announcement about the behaviors of the nodes it observes. This public announcement can be implemented by having the nodes exchange the behaviors of nodes they observe through broadcasting. The intersection of these announcements can be utilized to identify the deviating node. At the end of each stage game, the nodes report either no nodes deviate or the identity of the deviating node. Since these announcements can be exchanged in a relatively low frequency and only to the related nodes, the communication overheads are limited. Under this local observability assumption, the following theorem inspired by the Folk Theorem for privately monitoring with communication [15] is proposed

Theorem 3: Suppose $\mathbf{V}^\dagger$ has N dimensionality (full dimensionality), where N is the number of nodes in the network. If every node i is monitored by at least two other nodes, this implies the following:

1. If node i participates in the routes that have only 2 hops, then $deg_{in}(i) \geq 2$ is sufficient.
2. If node i participates in the routes which one of the routes has only 2 hops, then $deg_{in}(i) \geq 3$ is sufficient.
3. If node i participates in the routes which have more than 2 hops, then $deg_{in}(i) \geq 4$ is sufficient.

Also, there always exists a node that can punish the deviating node, i.e.,

$$deg_{out}(i) > 0, \ \forall i. \tag{22}$$

Moreover, the monitoring nodes can exchange the observations. Then, for every v in the interior of $\mathbf{V}^\dagger$, there exist $\underline{\delta} \in (0,1)$, such that for all $\delta \in (\underline{\delta}, 1)$, $v = (v_1, \cdots, v_N)$ is an equilibrium of an infinitely repeated game in which node i's average utility is v_i.

Proof: Suppose there exist ε, δ and punishment period T_i such that (14) holds and

$$\sum_{t=0}^{\max_i\{T_i\}-1} \delta^t \max_i \{ \max_{(\alpha,\alpha')} \left(v_i(\alpha) - v_i(\alpha') \right) \} < \sum_{t=\max_i\{T_i\}}^{\infty} \delta^t \varepsilon, \tag{23}$$

then the following rules of the game (Condition I to III) achieves the equilibrium when $deg_{in}(i) = 2,\ \forall i$.

Condition I: If there is no announcement of the deviating nodes

a. If the previous stage is in cooperating state, continue the cooperating state.

b. If the nodes play the following strategy in the previous stage

$$\left(U^{(1)}, \cdots, U^{(k-1)}, U^{(k)} - \varepsilon, U^{(k+1)}, \cdots, U^{(N)}\right)$$

for $k \in \{1, \cdots, N\}$, continue the previous state.

c. If the previous stage is in punishing node k state and the punishment has not ended, then continue the punishing. Otherwise, switch to strategy that results in

$$\left(U^{(1)}, \cdots, U^{(k-1)}, U^{(k)} - \varepsilon, U^{(k+1)}, \cdots, U^{(N)}\right).$$

Condition II: If node j is incriminated by both of its monitors j_1 and j_2

a. If the previous stage's strategy is either in the following states: punishing node j, implementing $\left(U^{(1)}, \cdots, U^{(j-1)}, U^{(j)} - \varepsilon, U^{(j+1)}, \cdots, U^{(N)}\right)$, implementing $\left(U^{(1)}, \cdots, U^{(j)} - \varepsilon, \cdots, U^{(l)} - \varepsilon, \cdots, U^{(N)}\right)$, for some $l \neq j$, or in implementing $\left(U^{(1)}, \cdots, U^{(l)} + \varepsilon, \cdots, U^{(j)} - \varepsilon, \cdots, U^{(N)}\right)$, for some $l \neq j$, then start the punishment stage for punishing node j.

b. If the previous stage's strategy is in punishing node j_1, then switch to the strategy that results in $\left(U^{(1)}, \cdots, U^{(j_2)} + \varepsilon, \cdots, U^{(j)} - \varepsilon, \cdots, U^{(N)}\right)$. The similar argument is applied to increase node j_1's utility by ε when node j_2 is punished in the previous stage.

Condition III: If there is any inconsistent announcement by node j_1 and j_2. We note that the inconsistent announcement happens when there are at least two announcements of the deviation node, but the deviation nodes in the announcements are different.

a. If the previous state is punishing node j_1 or node j_2, then restart the punishment stage.

b. Otherwise, implement $\left(U^{(1)}, \cdots, U^{(j_1)} - \varepsilon, \cdots, U^{(j_2)} - \varepsilon, \cdots, U^{(N)}\right)$.

In the above rules, we consider three different conditions, namely when no announcement of deviating node (Condition I), when the announcements are consistent (Condition II), and when the announcements are inconsistent (Condition III). Then we discuss the different strategies for different states within each Condition. We note that only the nodes whose packets are forwarded by node j have the potential ability of detecting the deviation of node j. The above game rule ensures that if every nodes in the network are monitored by at least two other nodes and there always exist nodes to punish the deviating node, then any $v \in \mathbf{V}^\dagger$ can be realized.

If both the monitors (node j_1 and node j_2) of node j incriminate node j, then node j is punished in the similar way to the punishment in Theorem 1. The deviator (node j) is punished for a certain period of time if the previous state is in one of the following states: punishing node j state (this implies that the punishment stage will be restarted), finished punishing node j state (i.e. in the state with utility function as $U^{(1)}, \cdots, U^{(j-1)}, U^{(j)} - \varepsilon, U^{(j+1)}, \cdots, U^{(N)}$), after penalizing nodes that make inconsistent announcements (i.e. in state with utility $U^{(1)}, \cdots, U^{(k)} - \varepsilon, \cdots, U^{(l)} - \varepsilon, \cdots, U^{(N)}$), where node k and l are the nodes that previously make inconsistent announcements, or in state with utility $U^{(1)}, \cdots, U^{(l)} + \varepsilon, \cdots, U^{(k)} - \varepsilon, \cdots, U^{(N)}$. In all these states, the deviator (node j) will be punished for a certain period of time (Condition IIa). However, if the previous state is in punishing node j_1, then the system switches to strategy that results in $U^{(1)}, \cdots, U^{(j_2)} + \varepsilon, \cdots, U^{(j)} - \varepsilon, \cdots, U^{(N)}$ (Condition IIb). This strategy gives additional incentives ($U^{(j_2)} + \varepsilon$) for node j_2 to punish to node j. Obviously, node j_1 has the incentive to announce if node j deviates, since this announcement will end node j_1 punishment. Because of the possible early termination of the punishment period, node j_1 also has the incentive to wrongly incriminate node j, this particular case will be prevented by Condition IIIa. Condition IIb is also used to avoid the situation where node j_2 lies on its announcement even though it observes that node j deviates. This condition will become obvious as we discuss the Condition III.

Next, we consider the case where there are incompatible announcements. We note that incompatible announcements imply that there are two nodes or two groups of nodes that make different announcements on the deviation. These announcements can be in the forms of either node j is only incriminated by one of the nodes (a group of nodes) or two different nodes are incriminated by two other nodes (two other groups of nodes). When there are incompatible announcements about node j (Condition III) and the previous state is not in punishing node j_1 or j_2, the nodes that make incompatible announcements will be penalized and they will receive utility $U^{(j_i)} - \varepsilon$ for $i = 1, 2$ (Condition IIIb). In the case when node j_1 is being punished in the previous stage, the Condition IIIa prevents node j_1 from falsely accusing node j. Condition IIIa and Condition IIIb are sufficient to avoid lying in announcement. However, including Condition IIIa creates the situation where node j_2 enjoy punishing node j_1. This means that when node j_1 is being punished and in the case node j has really deviated, node j_2 has the incentive to lie in its announcement and announces that no nodes is deviating. This problem is solved by Condition IIb that gives additional reward for node j_2 to tell the truth and punish node j. Moreover (23) implies that this additional reward for node j_2 outweighs the benefit from punishing node j_1. (23) can be thought as the incentives for the monitoring nodes to punish the deviating node when the announcements are inconsistent.

Previous arguments ensure that if every nodes in the network are monitored by at least two other nodes, then any feasible $v \in \mathbf{V}^\dagger$ can be realized. Next, we analyze the three cases listed in Theorem 3. In the first case, if all routes that node i participates have only 2 hops, and $deg_{in}(i) \geq 2$, this implies that every node can be perfectly monitored by two or more nodes. It is obvious that the above game rules can be applied directly. In the second case when node i participates in routes with one of the routes of exactly 2 hops, and $deg_{in}(i) \geq 3$, both the announcement from the source of the 2-hop route and the aggregate announcements from the sources of the rest of the routes serve as the final announcements. We note that

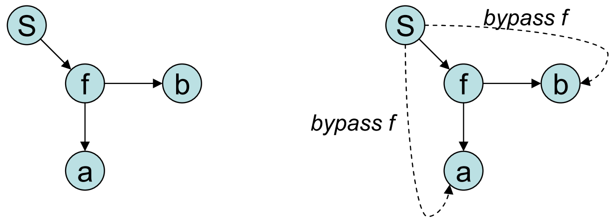


Fig. 3. Suppose the victim node, S, is in the edge of the network and every transmission coming from node S should go through node f. Suppose node f deviates and blocks the announcement from S. Node S can increase the transmission power to bypass node f to broadcast the announcement.

the intersection of the aggregate announcements will do the incrimination on a certain node. The node that does not tell the truth can be determined by majority voting method. Finally, for the case where node i participates in the routes which have more than 2 hops and $deg_{in}(i) \geq 4$, the sources can form two groups and use the previous game of rule. The lying node will be detected using majority voting. In summary, any potential deviation in the network satisfying the conditions of Theorem 3 can be detected. Moreover, the game rules guarantee that any feasible rational utilities can be enforced. ■

We note that from the announcement forwarder perspective, it faces two scenarios, namely either the announcement contains negative information about the forwarder itself or it contains negative information about the other nodes. In the first case, the forwarding node may not forward the announcement, however, even though that node itself does not forward the announcement, there is only a small probability that the announcement does not go through the whole network as illustrated in Figure 3. Moreover, the condition that every node is monitored by at least 2 nodes indicates that the illustrated case is less probable. In the second case, the forwarding nodes do not have any immediate gain for not forwarding the announcement, i.e., the forwarder is indifferent of forwarding the announcement. However, the forwarding nodes are better-off to forward the truthful announcement in order to catch and punish the deviating node. Otherwise, the forwarding nodes may also become the victims of the deviation in the future. Moreover, the announcement consumes much lower energy compared to the packet transmission itself. Hence, by indifferent we meant, each node is better off while making a truthful announcement, which will consume just a small portion of the energy transmission rather than a bigger loss when it is deviated by the deviating node.

Based on different information structures, analyses in Section III-A and Section III-B guarantee that any individually rational utilities can be enforced under some conditions. However, the individual distributed nodes need to know how to cooperate, i.e. what the good packet forwarding probabilities are. In the next section, we describe the learning algorithms to achieve better utilities.

IV. Self-Learning Algorithms

From Section III, any Pareto dominant solutions better than one stage NE can be sustained. However, the analysis does not explicitly determine which cooperation point to be sustained. In fact, the system can be optimized to different cooperating points, depending on the system designer choices. For instance, the system can be designed to maximize the weighted sum of the average infinitely repeated game's utilities as follow

$$\overline{U}_{sys} = \sum_{i=1}^{N} w(i)\overline{U}_{\infty}^{(i)}, \quad \texttt{where} \quad \sum_{i=1}^{N} w(i) = 1. \tag{24}$$

In particular, when $w(i) = \frac{1}{N}, \forall i$, maximize the average utility per nodes is usually employed in network optimization

$$\overline{U}_{sys} = \frac{1}{N}\sum_{i=1}^{N} \overline{U}_{\infty}^{(i)}. \tag{25}$$

We use (25) as an example, but we emphasize that any system objective function can be incorporated into the learning algorithm in a similar way. From individual point of view, as long as the cooperation can generate a better utility than the non-cooperation, the autonomous node will participate. Moreover, any optimization other than the system optimization can be monitored by the other nodes as deviation. Consequently, the punishment can be explored in the future.

The basic idea of the learning algorithm is to search iteratively the good cooperating forwarding probability. Similar to the punishment design, we consider the learning schemes for different information availability, namely, the perfect observability and the local observability. In parallel with the system model in Section II, we consider the time-slotted transmission that interleaves the learning mode and the cooperation maintenance mode as shown in Figure 1. In the learning mode, the nodes search for better cooperating points. In the cooperation maintenance mode, nodes monitor the actions of other nodes and apply punishment if there is any deviation. In the learning mode, the nodes have no incentives to deviate since they do not know if they can get benefits. So they do not want to miss the chance of obtaining the better utilities in the learning mode. It is also worth mentioning that if a node deviates just before a learning period, it will still be punished in the following cooperation maintenance period. So the infinite repeated game assumption is still valid in this time slotted transmission system.

A. Self-learning under the perfect observability

Under the perfect observability information structure, every node is able to detect the deviation of any defecting node, and observe which nodes help forwarding others' packets. This fact implies that every node is able to perfectly predict the average efficiencies of other nodes and optimize the cooperating point based on the system criterion (25). The basic idea of the learning algorithm is to use the steepest-descent-like iterations. All nodes predict the average efficiencies of the others and the corresponding gradients. The detailed algorithm is listed as in Table I. Learning with perfect observability assumes the perfect knowledge of utility functions of all nodes in the network, and represents the best solution that any learning algorithm can achieve.

TABLE I

SELF-LEARNING REPEATED-GAME ALGORITHM UNDER PERFECT OBSERVABILITY

For node i: Given $\vec{\alpha}_{-i}$, small increment β, and minimum forwarding probability α_{min}
Iteration: $t = 1, 2, \cdots$ Calculate $\nabla\overline{U}_{sys}(\vec{\alpha}(t-1))$ Calculate $\vec{\alpha}(t) = \vec{\alpha}(t-1) - \beta\nabla\overline{U}_{sys}(\vec{\alpha}(t-1))$ Select $\alpha_i(t) = \min\{\max\{[\vec{\alpha}(t)]_i, \alpha_{min}\}, 1\}$

TABLE II

SELF-LEARNING REPEATED-GAME ALGORITHM (FLOODING)

Initialization: $t = 0$ $\alpha_i^t = \alpha_0, \forall i$. Choose small increment ξ, η.
Iteration: $t = 1, 2, \cdots$ Calculate $U^{(i),t-1}(\alpha_i^{t-1})$ and $U^{(i),t-1}(\alpha_i^{t-1}+\xi)$, Calculate $\Delta U^{(i),t-1} = U^{(i),t-1}(\alpha_i^{t-1}+\xi)$ $-U^{(i),t-1}(\alpha_i^{t-1})$, For each i such that $\Delta U^{(i),t-1} > 0$, $\alpha_i^t = \alpha_i^{t-1} + \eta\frac{\Delta U^{(i),t-1}}{U^{(i),t-1}(\alpha_i^{t-1})}$, $\alpha_i^t = \max(\min(\alpha_i^t, 1), \alpha_{min})$. End when: No improvement. Keep monitoring the deviation Start punishment scheme if there is a deviation

B. *Self-learning under the local observability*

In this subsection, we focus on the learning algorithm with the information structure available under local observability. Under this condition, the nodes may not have the complete information about the exact utility of others. Based on this information structure, we develop two learning algorithms. The first algorithm is called *learning through flooding*. The second algorithm makes prediction of the other nodes' stage efficiency based on the flows that go through the predicting node. We called the second algorithm as *learning through utility prediction*.

1) Learning through Flooding: The basic idea of the learning algorithm is as follow. Since the only information the node can observe is the effect of changing its forwarding probability onto its own utility function. The best way for the nodes to learn the packet forwarding probability is to gradually increase the probability and monitor if the utility function becomes better. If the utility becomes better, the new forwarding probability will be employed. Otherwise, the old forwarding probability will be kept. The algorithm lets all nodes change their packet forwarding probabilities simultaneously. This can be done by flooding the instruction for changing the packet forwarding probability. After changing the packet forwarding probability, the effect propagates throughout the network. All nodes wait for a period of time until the network becomes stable. At the end of this period, the nodes obtain their new utilities. If the utilities are better than the original ones, then the new packet forwarding probabilities are employed. Otherwise, the old ones are kept. We note that the packet forwarding probability increment is proportional to the increase in the utility function: the nodes with higher increment in their utility functions increase their forwarding probability more compared to the nodes with lower utility increment. Here, we introduce the normalization factor $U^{(i),t-1}(\alpha_i^{t-1})$ (the utility before changing the forwarding probability) in order to keep the updates in forwarding probability bounded. The forwarding probability increment depends on small increment constant η and the normalization factor. The above process is performed until no improvement can be made. The detailed algorithm is shown in Table II.

We note that the time until the network is stable is defined as the time until all of the nodes do not observe fluctuations in their utility functions as the result of flooding/changing forwarding probabilities in the previous round. In practice, this waiting time can be either predefined or adjusted online as follow: Depending on the size of the network, a waiting period will be set in each node. If the node observes that its utility function fluctuates more than the preset period of time, that node can propose to prolong the preset time in the next round of flooding, otherwise the old preset waiting time is employed. When a node observes requests to prolong the waiting time, it sets the maximum of the broadcasted waiting times and its own waiting time as the current waiting time. In this way, nodes will wait until the effect of changing forwarding probability propagates to the whole network before the next flooding (changing of forwarding probability) happens. The maximum delay can also be set to keep the delay time bounded.

2) Learning with utility prediction: In this second approach, we observe that some of the routing information can be used to learn the system optimal solution (25). We assume that the routing decision has been made before performing the packet forwarding task. For instance, in the route discovery using Dynamic Source Routing (DSR) [12] algorithm without route caching, the entire selected route is included in the packet header in the packet transmission. The intermediate nodes use the route (in packet header) to determine to whom the packet will be forwarded. Therefore, it is clear that the transmitting node knows where the packet goes through, the relaying nodes know where the packet comes from and heads to, and the receiving node knows where the packet comes from. The nodes use this information to predict the utilities of others' nodes. We note that because not all nodes are involved in all of the flows in the network, the utility prediction may not be perfectly accurate. But from the simulation results, the performance degradation is minimal since only the nearby nodes matter.

The utility prediction is illustrated using an example shown in Figure 4, assuming $\mu_{S(r)} = 1$, $K = 1$, and $d(i,j) = 1$. We denote $U_i^{(j)}$ as the utility of node j predicted by node i. From the figure, node 1 receives flows from node 3, and node 4 and node 4 receives flows from node 1 and node 2. It is obvious that the flow from node 2 to node 4 is not perceived by node 1. Hence, the utilities of node 2 and node 3 predicted by node 1 are not the accurate ones. Similarly, the flow from node 3 to node 1 is not perceived by node 4. Therefore, $U_4^{(2)}$ and $U_4^{(3)}$ are not accurate. The accuracy of the prediction depends

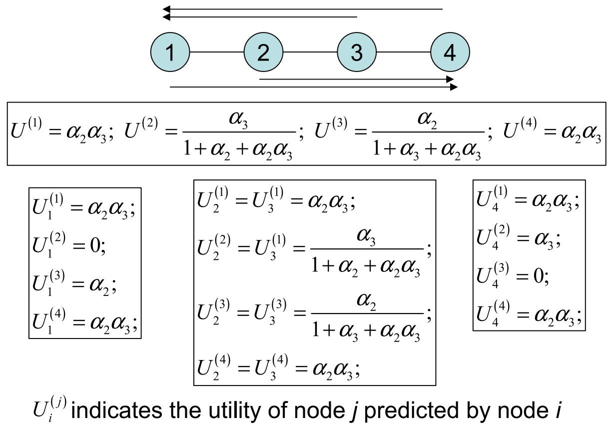


Fig. 4. Example for learning with utility prediction

on the flows. If all flows involving node i pass through node j then $U_j^{(i)}$ will be accurate and vice versa as illustrated in Figure 4. However, as we show by simulations the inaccuracy in the prediction does not affect the results of optimization too much.

Since the objective of the optimization is to achieve the system optimal solution (25), the best node i can do is to find the solution that minimizes the total average predicted utility function, which is

$$\min \tfrac{1}{N}\textstyle\sum_{j=1}^{N} U_i^{(j)}(\hat{\alpha}_i^{(1)},\cdots,\hat{\alpha}_i^{(N)}), \qquad (26)$$
$$\text{s.t. } \alpha_{min} \le \hat{\alpha}_i^{(j)} \le 1, \forall j,$$

where $\hat{\alpha}_i^{(j)}$ is the packet forwarding probability that node j should employ as predicted by node i. The detailed of the algorithm is presented as in Table III. The algorithm in Table III imitates the steepest-descent algorithm based on the predicted utility, where every node finds the gradient of the predicted utility and optimizes the predicted system utility (26). After obtaining $\{\hat{\alpha}^{(i)}\}$, each node sets its own packet forwarding probability as $\alpha_i^t = \hat{\alpha}_i^{(i)}$. We note that the optimization problem (26) can be done in a distributed manner, since the optimization does not require the global knowledge of the utility function. Each node does the optimization based on its own prediction and sets its packet forwarding probability according to the optimized predicted average utility.

Finally, we discuss how to handle the mobility of nodes. We note that the scheme will work well in moderate node mobility when the neighbors of each node do not change very often. Under this condition, the long-term relationship between nodes can be established by means of the repeated game and reputation announcement as described in Section III. As a result, the cooperation can be learned and enforced.

Obviously, the long-term relationship may be hard to establish in the case where there is a node that deviates in one part of the network, moves quickly to the other part of the network, deviates again and so on so forth. In this case, there are two possible solutions. First, when the node moves to a new place, in order for the node to transmit, some background check is necessary. This can be done in two ways: first, if the nearby nodes can share the announcement, then the neighbors of the node can obtain the announcement from the node's previous neighbors. And the new neighbors will know the reputation of this new node. The analogy of this case in the real life is when someone applies for a new job, the new employer always asks for the references from the old employers. And both employers can work harmoniously in a distributed manner. In the literature, the above idea is implemented in the trust establishment for ad hoc network such as [16].

TABLE III
SELF-LEARNING REPEATED-GAME ALGORITHM WITH UTILITY PREDICTION

Initialization: $t=0$ $\alpha_j^{(i),t}=\alpha_0, \forall i,j$. Choose small increment ζ.
Iteration: $t=1,2,\cdots$ For each node $j=1,\cdots,N$ Calculate $[\nabla_j^{(1)},\cdots,\nabla_j^{(N)}]=\left[\frac{\partial\sum_{n=1}^{N}U_j^{(n)}}{N\partial\hat{\alpha}_j^{(1),t}},\cdots,\frac{\partial\sum_{n=1}^{N}U_j^{(n)}}{N\partial\hat{\alpha}_j^{(N),t}}\right]$ Calculate $\alpha_j^{(i),t}=\alpha_j^{(i),t-1}+\zeta\nabla_j^{(i)}$ Set $\alpha_j^{(i),t}=\max(\min(\alpha_j^{(i),t},1),\alpha_{min})$. End when: No improvement and return $\alpha_j^{(i)}=\alpha_j^{(i),t},\forall i,j$. Keep monitoring the deviation, and go to punishment scheme whenever there is a deviation.

The other solution is by increasing the sampling of the learning algorithm. As long as the node mobility does not change the relationship between neighboring nodes drastically, the effect of mobility to the learning algorithm can be leveraged by putting more frequent learning period in the slotted transmission as in Figure 1. This case is similar to tracking non-stationary channel; the faster the channel changes the more frequent the training sequence transmission is required.

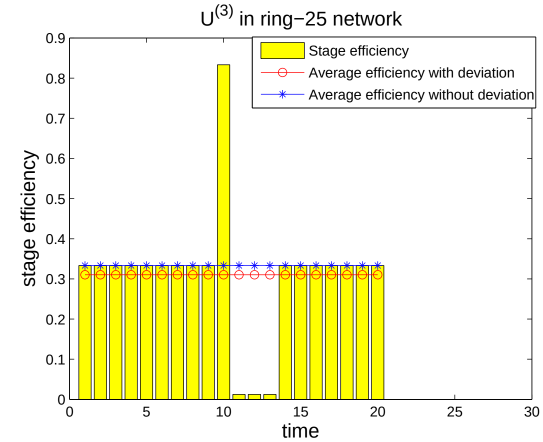
V. SIMULATION RESULTS

To investigate effectiveness of our proposed framework, we perform simulations with the following settings. We generate two networks with 25 nodes: the ring-25 network and random-25 network. The ring-25 network consists of 25 nodes that are arranged in a circle with radius $1000m$. The random-25 network consists of 25 nodes that are uniformly distributed in the area of $1000m \times 1000m$. We define the maximum distance d_{max}, such that two nodes are connected if the distance between two nodes is less than d_{max}. We select the maximum distance between two nodes to ensure connectivity of the whole network. In the ring-N network, the angle separation between two neighboring nodes is $\frac{2\pi}{N}$. And, the distance between two neighboring nodes is $2r\sin(\frac{2\pi}{2N})$, where r is the radius of the circle. In particular, the maximum distance for the ring-25 network can be calculated as $2000\sin\left(\frac{2\pi}{50}\right)m = 250.7m$. In the random-25 network, the maximum distance between two nodes is $350m$ to ensure connectivity of the whole network with a high probability.

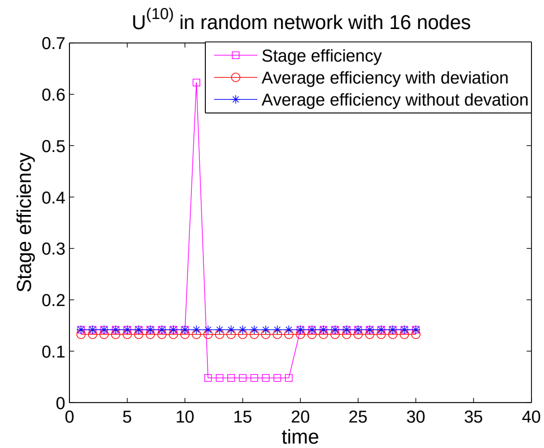
We also define the flows as source-destination (SD) pairs. We assume that the routing decision has been made before performing packet forwarding optimization. The shortest path routing is employed in the simulations. In the random-25 network, we vary the number of SD pairs. When there are traffic flows from all nodes to all other nodes, we called this traffic as dense flow that implies that each node has packets destined to the rest of nodes in the network. Obviously, the dense flow has $N \times (N-1)$ SD pairs in the N-node network. When the total flow is less than the dense flow, the SD pairs are determined randomly. In the ring-25 network, the number of SD pairs is defined in the following way. The $(K \cdot N)$ SD pairs are obtained when every node i sends packets to nodes $(\{mod(i+2, 25), \cdots, mod(i+K+1, 25)\}$. For instance, 25 SD pairs are obtained when every node i transmits packets to node $mod(i+2, 25)$, 50 SD pairs are obtained when every node i sends packets to nodes $\{mod(i+2, 25), mod(i+3, 25)\}$, etc. The rest of the simulation parameters are given as follows, transmission rate of source i as $\mu_i = 1$, $\forall i$, transmission constant $K = 1$, distant attenuation coefficient $\gamma = 4$. We compare three learning algorithms according to the information availability. The parameters for the learning algorithms are listed as follows $\beta = 0.05$, $\xi = 0.001$, $\eta = 1.0$, and $\zeta = 0.05$. The minimum forwarding probability is set to be $\alpha_{min} = 0.1$ and the maximum forwarding probability is set to be $\alpha_{max} = 1$. Finally, all algorithms are initiated with $\alpha_0 = 0.5, \forall i$. We note that in the following simulations, we employ the average efficiency per node defined in (25) as our performance metric.

Figure 5(a) shows the average efficiency of the deviation node in the ring-25 network when the number of source-destination is 75 with the discounted factor $\delta = 0.9$. In the figure, node 3 deviates at time instant 10. This deviation causes the stage efficiencies of node 1, 2 and 25 become lower. From the route, node 1, node 2 and node 25 suspect that nodes in $\{2, 3, 4\}$, $\{3, 4, 5\}$ and $\{1, 2, 3\}$ are deviating, respectively. The nodes in the network know that node 3 is consistent to be incriminated for deviation and start the punishment stage (Here, the punishment period is set to 3). The punishment scheme results in lower average stage efficiency as described in Figure 5(a). From the figure, the average efficiency without deviation is better than the average efficiency with deviation. It is clear that it is better off for node 3 to conform to the previously agreed cooperation point. As the result, no node wants to deviate, since the deviation results in worse average efficiency. Similarly, Figure 5(b) shows the average utilities of deviating node and other nodes in the random network with 16 nodes with the discounted factor 0.9. At time instant 11, node 10 in the network deviates. At the next time instant, all related nodes that detect deviation exchange the list of the incriminated nodes. The consistent incriminated node (in this case node 10) is punished for a certain period of time (in this figure, 8 period of time). From the figure, it is clear that node 10 will have higher average efficiency when it conforms. So from Figure 5(a) and Figure 5(b), the proposed repeated game can enforce the cooperation among autonomous greedy nodes.

Figure 6 and Figure 7 show the learning curves for the proposed self-learning repeated-game scheme for the ring-25 network and the random-25 network, respectively. In the figures, we compare the optimal solution, learning with perfect



(a) Ring network



(b) Random network

Fig. 5. Punishment of repeated game in the ring network and the random network

observability, learning with flooding, and learning with utility prediction. In Figure 6, all of the algorithms achieve the system optimal value when the source-destination pairs are 100, 200, and 275. The learning with perfect observability and the learning with utility prediction have approximately the same convergence speed. The learning with flooding converges slower, since the learning with flooding does the trial-and-error to find the better forwarding probabilities. This unguided optimization although requires minimal information has the inferior convergence speed. Figure 7 shows the learning curves of the proposed algorithms for random-25 network with different source-destination pairs. One can observe that the learning with utility prediction achieves very close efficiency per node compared to the optimal solution and learning with perfect observation. In contrast, the learning with flooding achieves inferior efficiency per node.

Figure 8(a) shows the learned average efficiency per node for the various algorithms with different traffic flows in the ring-25 network. The efficiency becomes lower as the number of source-destination pairs become larger. This can be explained as follows. Because of the symmetric property of the utility functions, the local optimal forwarding probabilities for all nodes are the same. It can be easily shown that the local optimal forwarding probabilities in the ring-25 network is 1

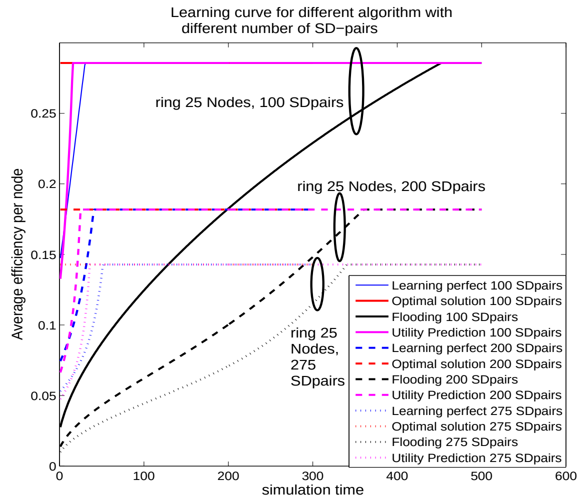


Fig. 6. Learned average efficiency per node for different traffic loads in the ring network

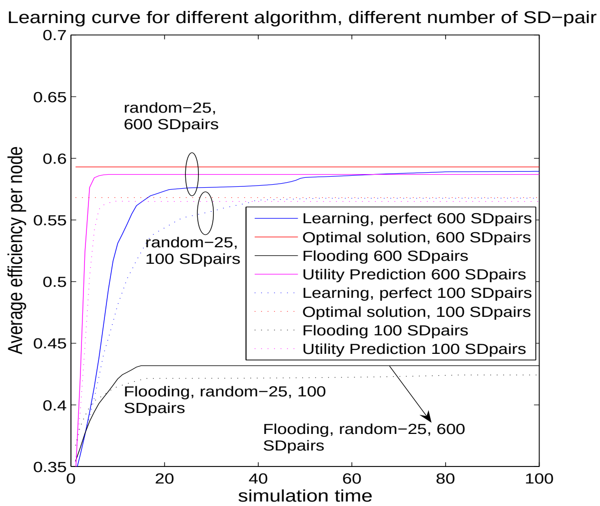
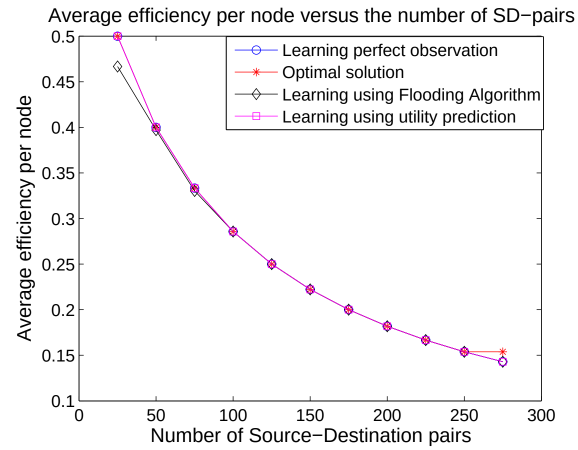
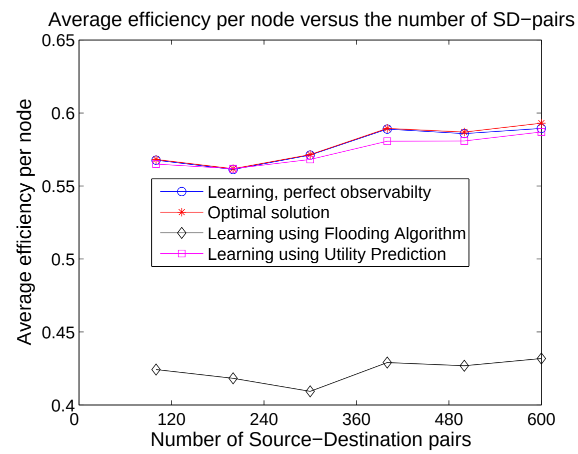


Fig. 7. Learned average efficiency per node for different traffic loads in the random network



(a) Ring network



(b) Random network

Fig. 8. Average efficiency per node for different traffic loads in the ring network and the random network

for all nodes[2]. Therefore, the larger the number of source-destination pairs, the more packets a node needs to forward and the higher value of the denominator of the stage utility function in (7). As the result, the average efficiency per node decreases as the number of source-destination increases. Using simple calculation, it can be shown that the average efficiency per node decays is $\frac{N_{sd}/N}{(N_{sd}/N+0.5*(N_{sd}/N+1)*(N_{sd}/N)}$, where N_{sd} is the number of source-destination pairs. In Figure 8(a), all learning algorithms perform similarly for the different numbers of source-destination pairs.

Figure 8(b) shows the achievable efficiency per node after the learning algorithms converge for different numbers of source-destination pairs in the random-25 network. We observe that the learning with utility prediction achieves very close efficiency compared to the learning with perfect observation and the optimal solution. The learning with flooding achieves lower efficiency per node, but still achieves much better efficiency compared to the Nash Equilibrium. In average, the learning with utility prediction achieves around 99.2% of the efficiency achieved by the optimal solution. In contrast, the learning with flooding achieves more than 73.18% of the optimality.

Comparing Figure 8(a) and 8(b), we can see that the learning with flooding performs well in the ring-25 network but inferior in the random-25 network. The reason for this phenomenon is that in the ring-25 network, the utilities of all nodes are symmetric and optimizing the system criterion (25) results in the same average efficiency in each node. Since the learning with flooding tries to increase its node's efficiency by changing its own forwarding probability synchronously, this iteration will finally reach the point where all nodes' efficiencies are the same due to the symmetric structure of the network. This solution is coincidentally the same as the solution of the system criterion (25) optimization. In contrast to the ring-25 network, the utility functions for each node are highly asymmetric in the random-25 network. In this case, the node that firstly reaches a better solution will not change its forwarding probability, even though changing its forwarding probability results in slightly lower efficiency in that particular node but increases the other nodes' efficiencies significantly.

[2]This is not true in the random network in general.

TABLE IV

NORMALIZED AVERAGE EFFICIENCY PER NODE FOR DIFFERENT NODES IN THE RANDOM NETWORK WITH DENSE TRAFFIC

Number of nodes	9	16	25	36	49	64	81
Average efficiency per node (Optimal solution)	0.7438	0.7581	0.5930	0.5574	0.5629	0.5316	0.4916
Normalized learning perfect observability	99.63%	99.91%	99.39%	100%	100%	100%	99.94%
Normalized learning using flooding	84.79%	71.45%	72.81%	65.36%	68.56%	58.21%	59.40%
Normalized learning using utility prediction	100%	97.91%	98.98%	99.27%	96.59%	99.88%	96.70%

Due to this greedy and unguided optimization, the learning with flooding achieves inferior average efficiency per node, compared to the learning using utility prediction which obtains information from routing information and performs better learning.

Next, we investigate the performances of the learning algorithms in the dense flow with different number of nodes in the random network. Table IV shows the average efficiency per node (25) for different sizes of the network normalized with the average efficiency obtained by the optimal solution. We can observe that as the number of nodes increases, the optimal average efficiency per node decreases. This is because the total power required for self-transmission and packet-forwarding increases much faster compared to the successful self-transmission power, as the number of nodes increases. Therefore, the stage utility for each node (7) decreases as the number of nodes increases in the dense flow. As the result, the average efficiency per node decreases as the node increases. We also observe that the learning with utility prediction achieves $96\% \sim 100\%$ of the average efficiency per node achieved by the optimal solution for various sizes of the network. On the other hand, the learning with flooding achieves $60\% \sim 85\%$ of the average efficiency obtained by the optimal solution. We note that the learning using flooding achieves lower efficiency as the number of nodes is larger, this is due to the unguided optimization. As the number of nodes becomes larger, it is more probable to get into the situation where only small portion of nodes have high efficiency but the rest have very low efficiencies. In contrast, the performance of learning using utility prediction slightly decreases but achieves a very close performance compared to the learning with perfect observability for various sizes of the network as shown in Table IV. The decrease is because as the number of nodes becomes larger, the utility prediction becomes less accurate.

VI. CONCLUSIONS

In this paper, we propose a distributed mechanism for enforcing and learning the cooperation points among selfish nodes in wireless networks. Our proposed scheme consists of a repeated-game framework to enforce cooperation and learning algorithms to search for better cooperation points. From the analysis and simulations, we show that our proposed framework is very effective to enforce cooperation among greedy/selfish nodes. In practice, selfish nodes with local information may not know how to cooperate even though they are willing to do so. We propose learning algorithms to guide the distributed nodes to find better cooperating points. Depending on the information structures, the proposed learning algorithm by flooding and with utility prediction achieve $60\% \sim 85\%$ and $96\% \sim 100\%$ of the efficiency that is obtained by the optimal solution with global information and centralized optimization.

REFERENCES

[1] G. Owen, *Game theory*, 3rd ed. Academic Press, 2001.

[2] D. Fudenberg and J. Tirole, *Game theory*, MIT Press, Cambridge, MA, 1991.

[3] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring, “Modelling incentives for collaboration in mobile ad hoc networks”, *Performance Evaluation 57*, pp. 427-439, 2004.

[4] S. Zhong, J. Chen, and Y. R. Yang, “Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks”, in *Proceedings of IEEE Annual IEEE Conference on Computer Communications (INFOCOM)*, pp.1987-1997, San Fransisco, March 30-April 03,2003.

[5] S. Marti, T. J. Giuli, K. Lai, and M. Baker, “Mitigating routing misbehaviour in mobile ad hoc networks”, in *Proceedings of ACM/IEEE Annual International Conference on Mobile Computing and Networking (Mobicom)*, pp.255-265, Boston, MA, August 2000.

[6] S. Buchegger and J-Y. Le Boudec, “Performance analysis of the CONFIDANT protocol (cooperation of nodes-fairness in dynamic ad-hoc networks)”, in *Proceedings of 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausannae, Switzerland, pp.80-91, June 2002.

[7] E. Altman, A. A. Kherani, P. Michiardi, and R. Molva, “Non-cooperative forwarding in ad hoc networks”, in *Proceedings of Networking 2005, 4th IFIP International Conferences on Networking, May 2 - 6, 2005, Waterloo, Canada - Springer LNCS Vol 3462, 2005*, May 2005.

[8] Z. Han, C. Pandana, and K. J. R. Liu, “A self-learning repeated game framework for optimizating packet forwarding networks”, in *Proceedings of IEEE Wireless and Communications and Networking Conference (WCNC)*, pp. 2131-2136, New Orleans, LA, March 2005.

[9] R. J. La and V. Anantharam, “Optimal routing control: repeated game approach”, *IEEE Transactions on Automatic Control*, vol.3, no.3, pp 437-450, March 2002.

[10] A. B. MacKenzie and L. A. DaSilva, *Game Theory for Wireless Engineers*, Morgan and Claypool Publishers, 2006.

[11] Z. Han, Z. Ji, and K. J. R. Liu, “Dynamic distributed rate control for wireless networks by optimal cartel maintenance strategy”, *IEEE Global Telecommunications Conference (Globecom)*, pp. 3742-3747, Dallas, TX, December 2004.

[12] D. Johnson, D. Maltz, Y. C. Hu, and J. Jetcheva, “The dynamic source routing protocol for mobil ad hoc networks (DSR)”, *IETF Internet Draft*, February 2002.

[13] M. Kandori, “Social norms and community enforcement”, *Review of Economic Studies*, vol.59, no.1, pp.63-80, 1992.

[14] D. Fudenberg and E. Maskin, “The Folk theorem in repeated games with discounting or with incomplete information”, *Econometrica*, vol.54, no.3, pp.533-554, May 1986.

[15] E. Ben-Porath and M. Kahneman, “Communication in repeated games with private monitoring”, *Journal of Economic Theory*, vol.70, pp.281-297, 1996.

[16] Y. Sun, W. Yu, Z. Han, and K. J. Ray Liu, “Information Theoretic framework of trust modeling and evaluation for ad hoc networks”, *IEEE Journal on Selected Areas on Communications*, vol.24, no.2, pp.305-317, February, 2006.