INPAC: An Enforceable Incentive Scheme for Wireless Networks using Network Coding

Tingting Chen and Sheng Zhong
Department of Computer Science and Engineering
The State University of New York at Buffalo
Buffalo, NY 14260-2000, U.S.A.
{tchen9, szhong}@cse.buffalo.edu

***Abstract*—Wireless mesh networks have been widely deployed to provide broadband network access, and their performance can be significantly improved by using a new technology called network coding. In a wireless mesh network using network coding, selfish nodes may deviate from the protocol when they are supposed to forward packets. This fundamental problem of packet forwarding incentives is closely related to the incentive compatible routing problem in wireless mesh networks using network coding, and to the incentive compatible packet forwarding problem in conventional wireless networks, but different from both of them. In this paper, we propose INPAC, the first incentive scheme for this fundamental problem, which uses a combination of game theoretic and cryptographic techniques to solve it. We formally prove that, if INPAC is used, then following the protocol faithfully is a subgame perfect equilibrium. To make INPAC more practical, we also provide an extension that achieves two improvements: (a) an online authority is no longer needed; (b) the computation and communication overheads are reduced. We have implemented and evaluated INPAC on the Orbit Lab testbed. Our evaluation results verify the incentive compatibility of INPAC and demonstrate that it is efficient.**

I. Introduction

Recently, wireless mesh networks [2], [8], [20], [21] have been widely deployed to provide broadband network access to schools, communities, and participants of various events. It is very challenging and highly important to improve the performance of wireless mesh networks so that the throughput scalability of such networks can meet the needs of different users. One way to achieve significantly better performance for wireless mesh networks is to apply a technique called *network coding* [7], [11], [14]–[16]. Unlike in conventional networks, in wireless networks using network coding, intermediate nodes do not store and forward the same packets as sent by the source node. In stead, intermediate nodes forward new coded packets computed by themselves from the packets they have received. Hence, the data is actually "mixed" at each intermediate node before it is forwarded by the intermediate node. This idea of mixing data at intermediate nodes takes advantage of the broadcast nature of wireless transmissions, and achieves great performance gains.

Many wireless mesh networks have user-contributed wireless devices as their nodes. Since users normally have their own interests, *economic incentives* become a crucial problem. A selfish or economically rational user may let her wireless device deviate from the communication protocol, as long as the deviation is beneficial to herself. However, this deviation may harm the network's performance, or even lead the network to stop functioning in the worst case. Therefore, we need to make the communication protocol *incentive compatible*, so that nodes have incentives to faithfully follow the protocol.

In this paper, we consider the incentive compatibility in wireless mesh networks using network coding. To be concrete, we assume that the network coding system used is MORE [7]. (In fact, our results can be adapted to some other network coding systems like MIXIT [15], through moderate modifications.) In a wireless mesh network using MORE, incentive compatibility is needed in at least two fundamental aspects: *routing* and *packet forwarding*. Here routing refers to the procedure of computing the number of transmissions each involved node should make for a data packet; packet forwarding refers to the procedure after routing that actually transmits packets from the source to the destination. These are two closely related, but completely different procedures. The incentive compatible routing problem in wireless networks using MORE has been studied by Wu et al. [28]. They propose a protocol that gives nodes incentives to honestly measure and report link loss probabilities in the routing procedure, and prove that following the protocol in the routing procedure is to the best interest of user nodes. Nevertheless, incentive compatible packet forwarding in the same kind of wireless networks has not received sufficient attention.

The main objective of this paper is to solve the incentive compatible packet forwarding problem in wireless mesh networks using a network coding system like MORE. We note that the incentive compatible packet forwarding problem has been studied extensively in the context of conventional wireless networks, i.e., wireless networks *not* using network coding. A lot of solutions have been proposed, e.g., [5], [6], [12], [19], [24], [30]. However, we emphasize that these existing solutions for conventional wireless networks cannot be used for wireless networks using MORE. For example, consider one such solution, Sprite [30]. In Sprite, in order to stimulate cooperation in packet forwarding, the source node makes payments to intermediate nodes along *the path to the destination* (which is usually the shortest path). For each packet originally sent by the source, the amount paid to each node is decided by whether *this particular packet* has been received by the destination, and whether the *next hop node* along the path reports having received this particular packet. In contrast, in a wireless mesh network using MORE, for at least three reasons we cannot use Sprite: (a) Packets are not forwarded along a predetermined *path* from a source to a destination. (b) Given an intermediate node, there is no well defined *next-hop node*. (c) Because intermediate nodes only transmit newly computed coded packets, it is nontrivial to decide the *correspondence relationship* between a packet sent by the source node and a packet transmitted by an intermediate node. Consequently, we have to look for a new solution for the incentive compatible packet forwarding problem in wireless mesh networks using MORE.

In wireless mesh networks using MORE, the problem of incentive compatible packet forwarding can be described as follows: Suppose that the routing procedure has already com-

puted the number of transmissions each node should make in order to forward a packet from a source node to a destination. We need to design an incentive scheme that stimulates nodes to faithfully follow the protocol and make exactly the number of transmissions computed by the routing procedure.

This problem is technically challenging in at least two aspects, as we briefly describe below. To address these technical challenges, we use novel techniques from game theory and cryptography, which we also briefly describe below.

The first technical challenge is in the economic aspect: It is nontrivial to find an economic method that gives nodes incentives to make the right number of transmissions—as far as we know, there is no existing method in game theory that we can directly apply. To address this challenge, we use game theoretic techniques to design a novel formula for paying packet forwarders. As long as this payment formula is enforced, we can guarantee that it is to the best interest of each forwarder to make the right number of transmissions.

The second technical challenge is in the security aspect: The enforcement of the payment formula mentioned above requires monitoring the transmissions each node has made to forward a packet, and this monitoring task must be carried out by some other node(s). However, if the latter node(s) do not report their results of monitoring correctly, we will not be able to calculate the right amount of payment that should be paid to the former node, and thus the former node may lose its incentives to follow the protocol. To address this challenge, we apply a combination of game theoretic and cryptographic techniques to allow nodes to punish each other for misbehavior like making incorrect reports. In this way, we can guarantee that our payment formula is properly enforced.

In summary, we have the following major contributions in this paper.

- We are the *first* to study the incentive compatible packet forwarding problem in the context of wireless mesh networks with network coding.
- To solve this problem, we use *novel* techniques to address the technical challenges and design an incentive scheme, *INPAC*. We formally prove that, if INPAC is used, then following the protocol faithfully is a subgame perfect equilibrium.
- To make INPAC more practical, we also provide an extension of INPAC in which two improvements are achieved: (a) an online authority is no longer needed; (b) the computation and communication overheads are reduced.
- To guarantee that INPAC can be effectively used in practice, we consider possible security attacks on INPAC and discuss defenses against them.
- We have completely implemented INPAC on the Orbit Lab testbed [23]. Our experimental evaluation results verify the incentive compatibility of INPAC and demonstrate that it is efficient.

The rest of this paper is organized as follows. In Section II, we describe the network model; since our work assumes an existing network coding system, MORE, we also briefly review MORE. In Section III, we design the INPAC basic scheme. In Section IV, we present a formal incentive analysis of the INPAC basic scheme. We present the INPAC extended scheme in Section V. We discuss two possible security attacks and the defenses against them in Section VI. Our evaluation results are described in Section VII. After reviewing related work in Section VIII, we conclude in Section IX.

II. Technical Preliminaries

Consider a wireless mesh network that has a set V of nodes. For $v_i, v_j \in V$, we denote by (v_i, v_j) the wireless link from node v_i to node v_j. Let $\epsilon_{i,j}$ be the loss probability of this link (v_i, v_j). So, if a packet is sent by node v_i, then node v_j can receive it with probability $1 - \epsilon_{i,j}$.

We assume that this wireless mesh network uses the network coding system MORE [7], and we will design our incentive scheme based on MORE. For completeness, we briefly review the packet forwarding procedure of MORE and some related terminologies.

Brief Review of MORE When a source node S sends packets to a destination node D, MORE works as follows:

Source Node: The source node S sends the packets in batches, where each batch has K *native* (i.e., uncoded) packets $\mathsf{NP}_1, \mathsf{NP}_2, \ldots, \mathsf{NP}_K$. S does not directly send these native packets; in stead, it sends *coded* packets, where each coded packet CP_j is a random linear combination of native packets: $\mathsf{CP}_j = \sum_{i=1}^{K} \mathsf{CV}_{ji}\mathsf{NP}_i$. The vector $\vec{\mathsf{CV}}_j = (\mathsf{CV}_{j1}, \mathsf{CV}_{j2}, \ldots, \mathsf{CV}_{jK})$ is called the *coding vector* of the coded packet CP_j. A MORE header is attached to each coded data packet, which contains the batch number, the coding vector, the source and destination addresses, and a list of (potential) forwarder nodes.

The list of forwarders is decided by S using the ETX metrics [10]. For $v_i, v_j \in V$, the ETX distance from v_i to v_j is denoted by $\mathsf{dist}(v_i, v_j)$. Intuitively, this means the expected number of transmissions to deliver a packet from v_i to v_j is $\mathsf{dist}(v_i, v_j)$. Given the destination node D, we say v_i is a *downstream node* of v_j if $\mathsf{dist}(v_i, D) < \mathsf{dist}(v_j, D)$. The source S chooses all its downstream nodes as forwarders, and orders them in the forwarder list according to their ETX distances to D.

Forwarder: When a node v_i hears a data packet, it checks whether it is in the packet's list of forwarders. Then, it checks whether the packet is *innovative* (i.e., linearly independent from all previous packets in the same batch that v_i has heard). If so, v_i makes a number of transmissions to forward this packet, where each packet transmitted by v_i is a random linear combination of all packets it has heard from this batch.

The number of transmissions v_i needs to make is precomputed in the routing procedure of MORE. We denote this number by $t_i^\star$. We assume that nodes follow the protocol faithfully in the routing procedure. Our main objective of this paper is to guarantee that each forwarder v_i will have incentives to make $t_i^\star$ transmissions for forwarding each data packet.

Destination Node: The destination D counts the number of innovative packets it has received. When it has received K innovative packets in the same batch, it sends an acknowledgement (which stops all nodes from transmitting packets in this batch) and decodes the received packets.

For further details of MORE, please refer to [7].

System Architecture The overall architecture of INPAC consists of a number of wireless nodes, on which MORE is implemented, and a central authority, called *Credit Clearance Center* (CCC). Note that having a central authority like the CCC is a standard assumption for incentive mechanisms in wireless networks (e.g., [3], [17], [26], [27]). We assume that the CCC issues a certificate to each node in the wireless mesh network and maintains an account of *credit* (i.e., *virtual currency*) for it, just like a central bank; so all the financial transactions between nodes will be cleared at the CCC. This authority can be either online or offline. For conceptual simplicity, in the basic scheme of INPAC as described in Section III, we assume an online

authority. In Section V, we present the INPAC extended scheme in which only an offline authority is needed.

When a node forward packets, it will receive payments from the source nodes of these packets as rewards. That is to say, forwarders get credit for their forwarding services and source nodes lose credit for receiving these services. In order to prevent nodes from making false claims about their forwarding services, we require that their downstream nodes submit reports to the CCC as evidence of such forwarding services. Details about these reports and how the CCC processes them are presented in Sections III and V.

III. Design of INPAC Basic Scheme

Given the network model and the system architecture, we now design an incentive scheme—the INPAC basic scheme, which stimulates cooperation in packet forwarding. Just like many existing incentive schemes in wireless networks (e.g., [3], [17], [26], [27], [31], [32]), INPAC uses *credit* to simulate cooperation. However, as we describe below, INPAC uses novel techniques that have never been used in existing schemes.

A. Main Ideas of the Design

To develop the main ideas of our INPAC basic scheme, let us consider a node v_i, which receives a packet that it is supposed to forward. Node v_i needs to decide the number of transmissions to make in order to forward this packet. Each of the transmissions v_i makes induces a cost c_i. However, the source node S will also make a payment to compensate v_i for the transmissions. For node v_i, the utility of making these transmissions equals the received payment minus the induced costs.

Recall that our objective is to guarantee that v_i has incentives to make exactly $t_i^\star$ transmissions, where $t_i^\star$ is computed by the routing procedure of MORE. To achieve this objective, we need to carefully design a payment formula, such that v_i maximizes the utility when it makes exactly $t_i^\star$ transmissions.

Payment Formula The first difficulty in designing the payment formula is that, in practice, no one except v_i itself can count precisely how many transmissions v_i actually makes; so, if the payment to v_i is based on the number of transmissions made by v_i, then there is no way to enforce the payment in reality. To sidestep this difficulty, we propose that node v_i should be paid in a constant amount p_i for *each packet received by at least one of its downstream nodes*. In this way, the incentive scheme can be enforced as long as the following two conditions are satisfied: (a) Every downstream node correctly reports the transmissions it has received from v_i. (b) There is a formula for calculating p_i, which does not need the number of transmissions actually made by v_i or any other node.

Now we develop a formula for calculating p_i under the assumption that every node correctly reports to the CCC the transmissions that it has heard as a downstream node. (After developing the formula for p_i, we will study the case in which this assumption does not hold.)

When v_i makes t_i transmissions, the expected amount of payment v_i receives is

$$\bar{p}_i(t_i) = (1 - \epsilon_i^{t_i}) p_i,$$

where

$$\epsilon_i = \prod_{\mathsf{dist}(v_h, D) < \mathsf{dist}(v_i, D)} \epsilon_{i,h} \tag{1}$$

is the probability that a packet sent by v_i is not received by any downstream node[1]. Hence, v_i's utility of making t_i transmissions is

$$\begin{aligned} \bar{u}_i(t_i) =& \bar{p}_i(t_i) - t_i c_i \\ =& (1 - \epsilon_i^{t_i}) p_i - t_i c_i, \end{aligned} \tag{2}$$

where c_i is v_i's cost of making one transmission. From the first order derivative of $\bar{u}_i$, we can easily find that $\bar{u}_i(t_i)$ is maximized when

$$t_i = \frac{\ln -\frac{c_i}{p_i \ln \epsilon_i}}{\ln \epsilon_i}.$$

Since our objective is that $\bar{u}_i(t_i)$ is maximized when $t_i = t_i^\star$, we need that

$$t_i^\star = \frac{\ln -\frac{c_i}{p_i \ln \epsilon_i}}{\ln \epsilon_i}.$$

Solving this equation, we get the formula for p_i:

$$p_i = \frac{c_i}{\epsilon_i^{t_i^\star} \ln \frac{1}{\epsilon_i}}, \tag{3}$$

Preventing Incorrect Reports The above derivations are under the assumption that downstream nodes correctly report the transmissions they have received. Thus we need additional measures to prevent downstream modes from cheating in their reports about transmissions.

There are two types of possible cheating in downstream nodes' reports: *overreporting* (i.e., reporting transmissions that they have not actually received) and *underreporting* (i.e., not reporting transmissions they have received).[2]

To prevent overreporting, we propose that, before any node v_i sends any data packet, v_i should attach its own signature on the batch number and the coding vector to the packet. When the downstream nodes report v_i's transmissions that they have received, they must present v_i's signatures to the CCC as evidence.

To prevent underreporting, we propose that v_i punishes any downstream node that underreports the transmissions from itself. Specifically, for any downstream node v_j, using the link loss probability $\epsilon_{i,j}$ and the number of transmissions v_i has made, node v_i can easily calculate the number of transmissions v_j should hear during a time interval.[3] Hence, by comparing this calculated number, with the number of transmissions v_j has reported to the CCC, node v_i can find out whether v_j has underreported transmissions from itself. If v_j has, then v_i punishes v_j by disallowing v_j to forward any future packets sent by v_i.

To implement this punishment, we propose that v_i encrypts its future data packets using a key unknown to v_j, but known by all other downstream nodes. Note that v_i's signatures on batch number and coding vector are parts of the encrypted cleartexts. In this way, even if v_j forwards these packets, it will not be able to replace v_i's signature with its own, and thus will not get paid for forwarding these packets.[4] Other downstream nodes are not

[1] Recall that the values of these $\epsilon_{i,h}$ are measured in MORE. We assume the measured values are correct, as guaranteed in [28].

[2] In fact, there is also a possibility that overreporting is mixed with underreporting, which can be easily prevented using our approaches for preventing overreporting and underreporting.

[3] We assume that the number of transmissions v_i makes is sufficiently large during the time interval, so that the number of transmissions v_j hears converges to its mathematical expectation.

[4] Node v_j may be able to figure out the data in some of these packets by looking at packets forwarded by other downstream nodes. However, in this case, forwarding the former packets is no more than forwarding the latter packets. Overall v_j still loses profits in forwarding some packets.

affected and still can forward these packets and get payments. (In Section III-C, we describe a key setup that satisfies the above requirement.) We stress that, with the encryptions and decryptions introduced by this punishment, our total overheads remain small (see Section VII for the experimental evaluation results), because we use a *symmetric key* cryptosystem.

Preventing Punishment Abuse Given the punishment power as described above, node v_i may punish a downstream node that does not make incorrect reports.To prevent such abuse, we propose that each node monitors its upstream nodes. If v_j finds that its upstream node v_i punishes itself while v_j itself has not made any incorrect report, v_j stops reporting any transmissions it has received from v_i. Consequently, v_i is "deterred" from punishing v_j unless v_j has underreported its transmissions.

So far we have intuitively explained our main ideas in the design of INPAC basic scheme. For precise and formal analysis of why these ideas can work, please see Section IV.

B. INPAC Basic Scheme

Putting together the ideas we have discussed in Section III-A, we obtain the INPAC basic scheme which stimulates nodes' cooperation in packet forwarding, as described below.

We assume that the communications between the CCC and any other node use a reliable protocol, such that lost packets are always retransmitted. We also assume that the source node S submits a signed copy of the forwarder list to the CCC, so that the CCC knows the upstream/downstream relationships among nodes.

The INPAC basic scheme consists of two parts: nodes' operations and the CCC's algorithm.

INPAC Basic Scheme – Regular Operations

▷ $batch\_no$: the batch number of a packet.
▷ $code\_vec$: the coding vector of a packet.
▷ ID_{v_i}: the identity of a node v_i.
▷ SIG_{v_i}: a node v_i's signature on $(batch\_no, code\_vec)$.

Source Node: Same as the source node's operations in MORE. In addition, the source node S attaches an INPAC header to each outgoing data packet. The INPAC header contains SIG_S.

Forwarders: When node v_i receives a data packet from an upstream node v_j for which v_i is in the forwarder list, v_i does the follows:

1) If the data packet is encrypted using a key known by v_i, v_i decrypts it; if the data packet is encrypted using a key unknown to v_i, v_i discards it and remembers ID_{v_j}.
2) Node v_i verifies SIG_{v_j}.
3) Node v_i checks whether the coding vector is linearly independent from the previous packets in the same batch sent by v_j. If so, v_i generates a new record $(ID_{v_j}, batch\_no, code\_vec, SIG_{v_j})$ and keeps it.
4) Node v_i checks whether the coding vector is linearly independent from *all* previous packets in the same batch received by v_i (i.e., whether it is *innovative*). If it is, then v_i makes transmissions as specified in MORE. But before making these transmissions, v_i replaces SIG_{v_j} with its own signature and encrypts the packet if in step 1 the packet was decrypted.

Destination Node: Same as the destination's operations in MORE.

Fig. 1. INPAC Basic scheme - Regular Operations on Packets

Nodes' Operations In the INPAC basic scheme, nodes have two types of operations: *regular operations* on data packets and *periodic operations*.

Fig. 1 lists the regular operations for processing a data packet.

Fig. 2 lists the three types of periodic operations of each node v_i. Note that each type of periodic operations may have a different cycle according to the system requirements.

INPAC Basic Scheme – Periodic Operations

1. *Submitting Report:* Node v_i submits a report to the CCC, which contains all the records v_i created in the most recent cycle.

2. *Monitoring Downstream Nodes:*
For each downstream node v_j, v_i compares the number of its own transmissions that v_j has reported to the CCC in the most recent cycle with the estimated number of transmissions that v_j should report. If v_i finds that v_j has underreported its own transmissions, v_i does the follows:

- Before forwarding each future packet, v_i encrypts $[SIG_{v_i}, payload]$, using key k_{-j} (see Section III-C for how to compute k_{-j});
- Node v_i replaces its locally stored value of $\epsilon_{i,j}$ with 1, and recalculates $t_i^\star$ using the algorithm in MORE.[4]

3. *Monitoring Upstream Nodes:*
Node v_i checks, for each upstream node v_j, whether v_j has ever encrypted packets using a key unknown to itself. If so, v_i stops making records for v_j's transmissions in the future.

Fig. 2. INPAC Basic Scheme - Periodic Operations

CCC's Algorithm After a node v_j submits a report to the CCC, the CCC processes the report as follows, in order to clear transactions:

1) Verify all signatures in the report.
2) Verify that all coding vectors for the same batch and the same sender are linearly independent, and that they are linearly independent from all coding vectors for the same batch and the same sender in previous reports submitted by v_j.
3) For each sender identity ID_{v_i} in the report, verify that v_i is an upstream node of v_j.
4) Mark the verified records as submitted by v_j and keep them.
5) For each record in the report, check whether its coding vector is linearly independent from all previous coding vectors in the same batch for which its sender has been paid. If so, pay its sender (say v_i) the amount of $p_i = \frac{c_i}{\epsilon_i^{t_i^\star} \ln \frac{1}{\epsilon_i}}$ from the account of the source node S and mark the record with "paid v_i".

C. Key Setup for Punishing Downstream Nodes

Suppose that node v_i would like to punish its downstream node v_j. We use the Akl-Taylor technique [1] to establish a key setup. This key setup allows v_i to compute a key k_{-j}, such that k_{-j} can be derived by any node except v_j. In this way, v_i can punish v_j by encrypting future packets using the symmetric key k_{-j}.

Let $N = Q_1 Q_2$ be an RSA modulus, where Q_1 and Q_2 are two large primes. Suppose that $k_0 \in Z_N^*$ is confidential to *all* nodes (i.e., no node knows k_0). We assume that each node v_i is

preloaded with a large prime P_i and $k_i = k_0^{P_i}$. Node v_i keeps k_i secret and makes P_i public.

The key k_{-j} for punishing v_j can be computed as

$$k_{-j} = k_i^{\prod_{h \neq i,j} P_h}.$$

It is easy to see that, for any $i' \neq j$,

$$k_{-j} = k_i^{\prod_{h \neq i,j} P_h} = k_0^{\prod_{h \neq j} P_h} = k_{i'}^{\prod_{h \neq i',j} P_h}. \quad (4)$$

Hence, any other node $v_{i'}$ ($i' \neq j$) can compute k_{-j} using Equation (4). However, it is infeasible for v_j to compute k_{-j}.

IV. Game Theoretic Analysis of INPAC Basic Scheme

In this section, we present a game theoretic model and formally analyze our INPAC basic scheme in this model.

A. Game Theoretic Model

We model the packet forwarding procedure of a particular session as a repeated game. The players of this game are the nodes that are required by the MORE protocol to forward the packets in this session. We assume that there are n players in total.

The game is divided into stages. In each stage ℓ, each node v_i chooses an action $a_{i,\ell}$, which is a tuple: $a_{i,\ell} = (t_{i,\ell}, \mathsf{PU}_{i,\ell}, \mathsf{PD}_{i,\ell})$. Here, $t_{i,\ell}$ is the number of transmissions v_i chooses to make for forwarding each packet in stage ℓ; $\mathsf{PU}_{i,\ell}$ ($\mathsf{PD}_{i,\ell}$, resp.) is the set of upstream (downstream, resp.) nodes v_i chooses to punish in stage ℓ. The utility of v_i in stage ℓ is

$$u_{i,\ell} = f_{i,\ell} \cdot (p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}_{h,\ell}} \epsilon_{i,h}^{t_{i,\ell}}) - c_i t_{i,\ell}),$$

where $f_{i,\ell}$ is the number of new packets that are received and need to be forwarded by v_i in stage ℓ.

As in the standard theory of repeated games [22], we consider a *discounting* total utility in the entire game. Let $\delta < 1$ be a constant—we call it the discount factor. The total utility of v_i in the entire game is

$$u_{i,\mathbf{total}} = \sum_{\ell=1}^{\infty} \delta^{\ell-1} u_{i,\ell}.$$

Intuitively, this means the player v_i has more interest in the current stage and the near future than in the far future.

In a repeated game, a history refers to a number of continuous stages starting from the beginning of the entire game, such that the actions of all players in all these stages have been chosen. The length of a history is defined as the number of stages in it. Given a history, the players can play the rest of the game, which constitutes a subgame. In each subgame, we can consider the utility of each player just as in the entire repeated game. For example, for a history H of length L, the utility of v_i in the subgame immediately following H is

$$u_i|_H = \sum_{\ell > L} \delta^{\ell-1} u_{i,\ell}.$$

[4]Setting $\epsilon_{i,j}$ to 1 reflects the fact that v_j does not report hearing packets from itself. The recalculation of $t_i^\star$ allows v_i to increase its number of transmissions when v_i finds one or more downstream nodes do not report hearing its packets. Technically, it is crucial to have this step in our protocol so that we can establish a subgame perfect equilibrium. Of course, we note that v_i might be making more transmissions than necessary to deliver packets in this case. However, this additional cost is not high and we get it only if some node deviates from the protocol. When the system converges to the subgame perfect equilibrium, this cost is not incurred. Note that, the per-packet payment p_i is *never* recalculated.

For the entire game, a strategy s_i for each player v_i specifies what action v_i should choose after each possible history. Clearly, once every player has determined its strategy, the utilities of all players are also fixed. Hence, for each player v_i, the total utility, the utility in any stage, or the utility in any subgame, is always a function of the profile of all players' strategies. Denote by s ($s = (s_1, \ldots, s_n)$) the profile of all players' strategies. We use $u_i|_H(s)$, to represent the utility of node v_i in the subgame immediately following history H, when the strategy profile s is used by the players.

We say a strategy profile $s = (s_1, \ldots, s_n)$ induces a *Nash equilibrium* in the subgame immediately following history H if for all v_i, for all $s_i' \neq s_i$,

$$u_i|_H(s_1, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n) \leq u_i|_H(s).$$

We say s is a *subgame perfect equilibrium* if s induces a Nash equilibrium in every subgame.

B. Incentive Analysis

In the game theoretic model we have presented, we can obtain the following theorem regarding the incentive compatibility of our INPAC basic scheme.

Theorem 1: The strategy profile in which all nodes follow the protocol faithfully is a subgame perfect equilibrium in the game.

Proof: Denote by $s^\star$ the strategy profile in which all nodes follow the protocol faithfully. Consider an arbitrary history H of length L. Suppose that $H = H_1 H_2 \ldots H_L$. When the strategy profile $s^\star$ is used, after history H, each node v_i makes $t_i^\star$ transmissions and punishes upstream nodes in the set PU_i and downstream nodes in the set PD_i, i.e., $s_i^\star(H) = (t_i^\star, \mathsf{PU}_i^\star, \mathsf{PD}_i^\star)$. Given our INPAC basic scheme, $\mathsf{PU}_i^\star$ and $\mathsf{PD}_i^\star$ are decided as follows:

$$\mathsf{PU}_i^\star = \{v_j | \exists \ell, 1 \leq \ell \leq L \wedge H_\ell = (a_1, a_2, \ldots, a_j, \ldots, a_n) \wedge a_j = (t_j, \mathsf{PU}_j, \mathsf{PD}_j) \wedge v_i \in \mathsf{PD}_j\};$$

$$\mathsf{PD}_i^\star = \{v_j | \exists \ell, 1 \leq \ell \leq L \wedge H_\ell = (a_1, a_2, \ldots, a_j, \ldots, a_n) \wedge a_j = (t_j, \mathsf{PU}_j, \mathsf{PD}_j) \wedge v_i \in \mathsf{PU}_j\}.$$

Now consider an arbitrary v_i. Let $s^\triangle$ be an arbitrary strategy profile that differs from $s^\star$ only in v_i's action immediately following history H. Suppose that $s_i^\triangle(H) = (t_i^\triangle, \mathsf{PU}_i^\triangle, \mathsf{PD}_i^\triangle)$. If we can show that $u_i|_H(s^*) \geq u_i|_H(s^\triangle)$ always holds, by One-Deviation Theorem [22], we get that $s^\star$ is a subgame perfect equilibrium.

First, we can calculate the utilities as follows:

$$u_i|_H(s^\star) = \sum_{\ell \geq L+1} \delta^{\ell-1} f_{i,\ell}^\star (p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}_{h,\ell}^\star} \epsilon_{i,h}^{t_{i,\ell}^\star}) - c_i t_{i,\ell}^\star), \quad (5)$$

$$u_i|_H(s^\triangle) = \sum_{\ell \geq L+1} \delta^{\ell-1} f_{i,\ell}^\triangle (p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}_{h,\ell}^\triangle} \epsilon_{i,h}^{t_{i,\ell}^\triangle}) - c_i t_{i,\ell}^\triangle), \quad (6)$$

where $f_{i,\ell}^\star$, $t_{i,\ell}^\star$, and $\mathsf{PU}_{h,\ell}^\star$ are the number of packets needed to be forwarded by node v_i, number of transmissions made by v_i for each packet needed to be forwarded, and the set of upstream nodes punished by node v_h, respectively, in stage ℓ

when the strategy profile $s^\star$ is used; correspondingly, $f^\triangle_{i,\ell}$, $t^\triangle_{i,\ell}$, and $\mathsf{PU}^\triangle_{h,\ell}$ are the counterparts when the strategy profile $s^\triangle$ is used.

Second, for all $\ell > L$, all upstream node v_h of v_i, clearly we have that

$$\mathsf{PD}^\star_{h,\ell} = \mathsf{PD}^\star_{h,L} \cup \bigcup_{\mathsf{dist}(v_{h'},D)<\mathsf{dist}(v_h,D)} \mathsf{PU}^\star_{h',L},$$

and that

$$\mathsf{PD}^\triangle_{h,\ell} \supseteq \mathsf{PD}^\triangle_{h,L} \cup \bigcup_{\mathsf{dist}(v_{h'},D)<\mathsf{dist}(v_h,D)} \mathsf{PU}^\triangle_{h',L}.$$

Since $\mathsf{PD}^\star_{h,L} = \mathsf{PD}^\triangle_{h,L}$ and $\mathsf{PU}^\star_{h',L} = \mathsf{PU}^\triangle_{h',L}$, the above two equations imply that

$$\mathsf{PD}^\star_{h,\ell} \subseteq \mathsf{PD}^\triangle_{h,\ell}.$$

Since v_i's number of received packets is determined by its upstream nodes' sets of punished downstream nodes, we have that, for all $\ell > L$,

$$f^\star_{i,\ell} \geq f^\triangle_{i,\ell}. \tag{7}$$

Third, we observe that, for all $h \neq i$,

$$\begin{aligned} & \mathsf{PU}^\star_{h,L+1} \\ = \ & \{v_j | \exists \ell, 1 \leq \ell \leq L \wedge H_\ell = (a_1, a_2, \ldots, a_j, \ldots, a_n) \\ & \wedge a_j = (t_j, \mathsf{PU}_j, \mathsf{PD}_j) \wedge v_h \in \mathsf{PD}_j\} \\ = \ & \mathsf{PU}^\triangle_{h,L+1}. \end{aligned}$$

Hence,

$$\begin{aligned} & p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\triangle_{h,L+1}} \epsilon_{i,h}^{t^\triangle_{i,L+1}}) - c_i t^\triangle_{i,L+1} \\ = \ & p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\star_{h,L+1}} \epsilon_{i,h}^{t^\triangle_{i,L+1}}) - c_i t^\triangle_{i,L+1}. \end{aligned} \tag{8}$$

For all $\ell > L+1$, since

$$\mathsf{PU}^\triangle_{h,\ell} \supseteq \mathsf{PU}^\triangle_{h,L+1} = \mathsf{PU}^\star_{h,L+1} = \mathsf{PU}^\star_{h,\ell},$$

we have that,

$$\begin{aligned} & p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\triangle_{h,\ell}} \epsilon_{i,h}^{t^\triangle_{i,\ell}}) - c_i t^\triangle_{i,\ell} \\ \leq \ & p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\star_{h,\ell}} \epsilon_{i,h}^{t^\triangle_{i,\ell}}) - c_i t^\triangle_{i,\ell}. \end{aligned} \tag{9}$$

Now, we define a function of a single variable $t^\triangle_{i,\ell}$ (for an arbitrary $\ell > L$):

$$g(t^\triangle_{i,\ell}) = p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\star_{h,\ell}} \epsilon_{i,h}^{t^\triangle_{i,\ell}}) - c_i t^\triangle_{i,\ell}. \tag{10}$$

From (10), we can easily obtain that

$$\begin{aligned} & \frac{\mathsf{dg}(\mathsf{t}^\triangle_{\mathsf{i},\ell})}{\mathsf{dt}^\triangle_{\mathsf{i},\ell}} \\ = & p_i \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\star_{h,\ell}} \epsilon_{i,h}^{t^\triangle_{i,\ell}} \ln \frac{1}{\prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\star_{h,\ell}} \epsilon_{i,h}} - c_i. \end{aligned}$$

Hence, $\frac{\mathsf{dg}(\mathsf{t}^\triangle_{\mathsf{i},\ell})}{\mathsf{dt}^\triangle_{\mathsf{i},\ell}} = 0$ if

$$t^\triangle_{i,\ell} = \log_{\prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\star_{h,\ell}} \epsilon_{i,h}} \frac{c_i}{p_i \ln \frac{1}{\prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\star_{h,\ell}} \epsilon_{i,h}}}.$$

Plugging the payment formula into the above, we get that $\frac{\mathsf{dg}(\mathsf{t}^\triangle_{\mathsf{i},\ell})}{\mathsf{dt}^\triangle_{\mathsf{i},\ell}} = 0$ if $t^\triangle_{i,\ell} = t^\star_{i,\ell}$.

Furthermore, from (10), we see that $\frac{\mathsf{dg}(\mathsf{t}^\triangle_{\mathsf{i},\ell})}{\mathsf{dt}^\triangle_{\mathsf{i},\ell}}$ always decreases. So, we have that $\frac{\mathsf{dg}(\mathsf{t}^\triangle_{\mathsf{i},\ell})}{\mathsf{dt}^\triangle_{\mathsf{i},\ell}} > 0$ if $t^\triangle_{i,\ell} < t^\star_{i,\ell}$, and that $\frac{\mathsf{dg}(\mathsf{t}^\triangle_{\mathsf{i},\ell})}{\mathsf{dt}^\triangle_{\mathsf{i},\ell}} < 0$ if $t^\triangle_{i,\ell} > t^\star_{i,\ell}$. Therefore, for all $t^\triangle_{i,\ell}$,

$$g(t^\triangle_{i,\ell}) \leq g(t^\star_{i,\ell}). \tag{11}$$

Combining (8)(9)(10)(11), we get that, for all $\ell > L$,

$$\begin{aligned} & p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\triangle_{h,\ell}} \epsilon_{i,h}^{t^\triangle_{i,\ell}}) - c_i t^\triangle_{i,\ell} \\ \leq \ & p_i(1 - \prod_{\mathsf{dist}(v_h,D)<\mathsf{dist}(v_i,D)}^{v_i \notin \mathsf{PU}^\star_{h,\ell}} \epsilon_{i,h}^{t^\star_{i,\ell}}) - c_i t^\star_{i,\ell}. \end{aligned} \tag{12}$$

From (5)(6)(7)(12), we get that

$$u_i|_H(s^\star) \geq u_i|_H(s^\triangle). \tag{13}$$

By equation (13), we know that $s^\star$ is a subgame perfect equilibrium. ∎

Theorem 1 tells us that there is a subgame perfect equilibrium in which all nodes follow the protocol. Clearly, in this subgame perfect equilibrium, each node makes exactly the number of transmissions required by MORE.

V. INPAC Extended Scheme

The INPAC basic scheme, which we have presented and analyzed in the previous sections, requires the CCC to always stay online. In this section, we present the INPAC extended scheme, which does not require the CCC to stay online. This extended scheme also has reduced computation and communication overheads compared with the basic scheme.

A. Main Ideas of Extended Scheme

Before we present our INPAC extended scheme in details, we intuitively explain the main ideas we use in our design of this extended scheme.

Using Offline CCC We no longer require nodes to clear transactions periodically when they are using the wireless mesh network; in stead, we allow nodes to use the wireless mesh network first, and clear the transactions only when they have high speed connections to the CCC. In this way, the mesh

network operator only needs to set up an Internet server as the CCC in order to satisfy our new requirement—this is a much easier task for the operator than maintaining an online authority.

Specifically, we let each node v_i periodically sign its records about each upstream node v_j's transmissions and submit the signed records to v_j itself, as receipts for transmissions from v_j. These receipts *may* allow v_j to get the corresponding payments when the transactions are cleared. As we have mentioned above, node v_j clears transactions only when it has a high speed connection to the Internet (i.e., to the CCC). At that time, the CCC checks each pair of $(batch\_no, code\_vec)$ in each receipt to see whether the coding vector is linearly independent from all coding vectors with the same batch number for which v_j has received payments. Node v_j receives a payment for this pair of $(batch\_no, code\_vec)$ only if the above condition is satisfied.

As in the basic scheme, each node v_i needs to monitor its downstream nodes for possible underreporting of its transmissions. Node v_i monitors a downstream node v_j by periodically checking the number of receipts it has received from v_j and comparing it with the number expected by itself.

Improving Efficiency It is easy to see that a large portion of the computation and communication overheads of our INPAC scheme comes from the generation, transmission, and processing of receipts. Consequently, to improve the efficiency of INPAC, we use a random sampling approach to significantly reduce the number of receipts that need to generated, transmitted, and processed.

Suppose that we would like to reduce the number of receipts to $\frac{1}{2^m}$ of the original, where m is a positive integer. We use a cryptographic hash function $\mathsf{Hash}()$ to help us do the sampling: For each node v_i, let x_i be a secret known by v_i and the CCC only. Whenever v_i receives a packet from its upstream node v_j, v_i needs to generate and submit a receipt for this packet only if the first m bits of $\mathsf{Hash}(x_i, ID_{v_j}, batch\_no, code\_vec, SIG_{v_j})$ are all zeros. Since $\mathsf{Hash}()$ can be viewed as a *random oracle* [4], each packet satisfies this condition with probability $\frac{1}{2^m}$.

This approach is secure and incentive compatible for the following reasons: (a) Upstream node v_j cannot cheat in this procedure. In particular, v_j cannot selectively transmit the sampled packets because it does not know x_i. (b) Node v_i cannot cheat to increase the number of generated receipts, because v_i cannot forge v_j's signature, which is part of the input to the hash function. (c) Node v_i cannot cheat to decrease the number of generated receipts, because then the cheating will be detected and punished by v_j, in a manner similar to the basic scheme.

B. INPAC Extended Scheme

Using the ideas we have just discussed, we obtain our INPAC extended scheme as follows.

Nodes' Regular Operations on Packets

Source Node: Same as the basic scheme.

Forwarders: When node v_i hears a packet from an upstream node v_j for which v_i is in the forwarder list, v_i does the follows:

1) A forwarder's regular operations in the basic scheme.
2) v_i checks: (a) whether the coding vector is linearly independent from the previous packets in the same batch sent by v_j and heard by v_i; (b) whether the first m bits of $\mathsf{Hash}(x_i, ID_{v_j}, batch\_no, code\_vec, SIG_{v_j})$ are all 0. If so, v_i makes a record $(ID_{v_j}, batch\_no, code\_vec, SIG_{v_j})$.

Destination Node: Same as the basic scheme.

Nodes' Periodic Operations

1) *Receipt Submission*: Each node v_i periodically signs its records about each upstream node v_j's transmissions and submits the signed records to v_j itself.When receiving the receipts, v_j verifies v_i's signatures and also verifies that all pairs of $(batch\_no, code\_vec)$ have indeed been transmitted by itself with v_i being a downstream node. Then, v_j keeps the receipts.
2) *Monitoring Downstream Nodes:* Each node v_i periodically counts, for each downstream node v_j, the number of packets sent by itself for which v_j have submitted receipts. Node v_i compares this number with $\frac{1}{2^m}$ of the total number of packets that v_j should have received from v_i. If v_j reports fewer packets than expected, then v_i punishes v_j using the same method as in the basic scheme.
3) *Monitoring Upstream Nodes:* Same as the basic incentive scheme.

Transaction Clearance

When a node v_i has a high speed connection to the CCC, v_i submits all the receipts it keeps. The CCC clears the transactions in a way similar to the basic scheme.

VI. Possible Attacks and Defenses

When our INPAC (basic or extended) scheme is used, security attacks may be launched by selfish or malicious nodes. Although the focus of this paper is incentives rather than security, for practical purposes, we still briefly consider two possible attacks and discuss the defenses against them.

A. Extra Signature Attack

A selfish forwarder node v_i may launch an attack on our protocol by putting some extra signed pairs of $(batch\_no, code\_vec)$ in the payload of a packet. For example, suppose v_i is going to send a packet that has batch number 001 and coding vector $(1, 1, 1)$. So, the pair $(001, (1, 1, 1))$ is in the MORE header of this packet, and the signature on this pair is in the INPAC header. Node v_i launches an attack by putting signed pairs $(001, (1, 2, 3))$ and $(002, (2, 1, 4))$ in the payload of this packet, in hope that the latter two signed pairs will also bring some payments to itself. Note that this attack will *not* work if all nodes hearing this packet follow the protocol, because nodes following the protocol should not take these signed pairs of $(batch\_no, code\_vec)$ (i.e, signed $(001, (1, 2, 3))$ and $(002, (2, 1, 4))$) from the payload and make records for them—they should make only one record for $(001, (1, 1, 1))$ from the MORE header and the corresponding signature from the INPAC header. Nevertheless, if a downstream node v_j hearing this packet does not follow the protocol, v_j may make records for these extra signed pairs and then submit the records to the CCC. In this case, v_i may get undeserved payments with the help of v_j.

We argue that the above attack is actually a colluding attack, because in this case v_i and v_j must have a prior agreement on the format of packet payload. One possible defense is that every node randomly samples a portion of packets it hears, to detect signed pairs of $(batch\_no, code\_vec)$ in the payload. If the attack is detected, it is reported to the network operator, who excludes the attacker nodes from the system and pursues liability against the owners of these nodes.

A better solution to this problem can be provided by a collusion-resistant incentive scheme. However, since collusion resistance is technically highly challenging, we leave this topic to future study.

B. Corrupted Data Attack

A forwarder node v_i may also launch an attack on our protocol by tampering with the payloads of data packets. When v_i receives a packet that it should forward, v_i can modify or remove part or all of the bits in the payload of this packet. This attack allows v_i to gets payments for forwarding packets while the destination does not receive the correct data in these packets.

We propose a simple defense against this attack: On one hand, when a node forwards a packet, it signs $(batch\_no, code\_vec, payload)$ rather than just $(batch\_no, code\_vec)$. Since this signature will be part of the receipt, the packet receipt can serve as an evidence of cheating if this node corrupts the payload. On the other hand, the source node S and the destination D should establish a secret key known to only S and D and protect the entire batch of data using a message authentication code (MAC). If a corrupted data attack is launched, the destination will detect the attack using the MAC. Then, the destination requests all forwarders to submit all their receipts to the source so that the source can determine who has corrupted the data. (This defense works for the basic scheme. If it is used for the extended scheme, then it finds the attacker node with a probability, which may deter the attacker.)

It is worth noting that this corrupted data attack is actually an *independent* security problem for network coding that exists *even if no incentive scheme is used*. It has been studied in, e.g., [11], [29]. Hence, we can also adapt existing solutions to our settings in order to defend against this attack.

VII. Evaluations

We completely implement INPAC and evaluate it on the Orbit wireless testbed [23]. Specifically, we carry out three sets of experiments:

- The first set of experiments are on the utilities of cheating nodes. The results show that a node reduces its own utility if it cheats in making transmissions, in reporting heard transmissions, or in punishing downstream nodes.
- The second set of experiments show that, starting from a network system where selfish wireless nodes have random (not necessarily cooperative) behaviors, INPAC makes the system quickly converge to a stable state. In this stable state, every node maximizes its own utility by faithfully following the protocol.
- The third set of experiments are on the computation and communication overheads. Our results show that INPAC is quite efficient.

Below we first describe the setups of all our experiments, and then present the detailed results for each set of experiments, respectively. Note that, since the basic scheme and the extended scheme are equivalent in terms of nodes' utilities and the system's convergence (except that the speeds of utility changes and system convergence depend on different parameters), *for the first two sets of experiments, we only present our results on the basic scheme*; the results on the extended scheme are similar. For the third set of experiments, we present our results for both the basic scheme and the extended scheme.

A. Setups of Experiments

From the Oribit radio grid testbed, we randomly select 30 nodes in the 20×20 grid. Fig. 3 shows the locations and IDs of these nodes. Each node has a 1-GHz VIA C3 processor with 512 MB of RAM and a 20 GB local hard disk, and is equipped with Atheros AR5002X Mini PCI 802.11a/b/g wireless card attached to an omni-directional antenna. Nodes are set to transmit at a power level of 20dbm and operate in the 802.11b mode with the bit rate 11Mbps. Softwares on each node include Linux Debian kernel v2.6.22, Mad-Wifi v0.9.3.3 [18], Click v1.6.0 [25], the MORE package [7], the Cryptopp Library 5.5.2 [9].

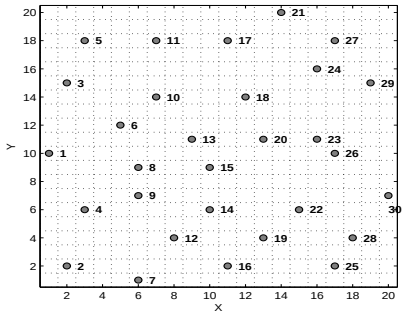

Fig. 3. Testbed Topology.

Before running the experiments, we use a module in MORE to measure the loss rate of each link, and find that the link loss rates vary between 17.07% and 100%. In MORE, we set the batch size to 32 packets, and the size of each packet is 1500 bytes. The minimum load threshold is set to 0.2 for MORE pruning module.

In the experiments on the basic scheme, we place the CCC on node 2 at location $(2, 2)$. Unless stated differently, nodes submit their reports to the CCC every 30 seconds. Each node checks, every 1 minute, whether any downstream node underreports its transmissions. In payment calculations, the cost of making each transmission is 1.

B. Nodes' Utilities and Cheating Behaviors

When a node deviates from the protocol in packet forwarding, it can have three types of basic cheating behaviors: changing the number of transmissions, underreporting upstream nodes' transmissions, and improperly punishing downstream nodes.[5] In this set of experiments, we study the nodes' utilities for each type of basic cheating behavior, respectively, and for a mixture of basic cheating behaviors.

We have 100 runs of each experiment described below. In each run we randomly choose two nodes as the source and the destination, to have a session of 120 seconds, in which the source node is always backlogged. Unless stated differently, in each run, we randomly select an involved node and compare the utilities it receives when it cheats and when it follows the protocol.

Changing Number of Transmissions We measure the per-batch utilities of nodes when they cheat by changing their t_i, the number of transmissions for forwarding each packet. When a node cheats, its t_i is randomly chosen between 0 and $2.0t_i^\star$, where $t_i^\star$ is the number of transmissions it should make when it follows the protocol.

Fig. 4 shows the scatter plot for utility comparison in the 100 runs. Each point represents the utilities of a randomly selected node in one run: The y-coordinate of the point is the node's utility when it changes its t_i and the x-coordinate is the same

[5]Actually a node may also cheat by overreporting upstream nodes' transmissions, but we ignore this possibility in our experiments because it is easily detected by the CCC through signature verifications.

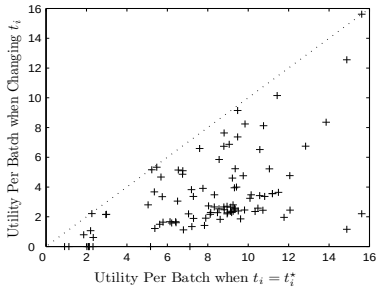

Fig. 4. Scatter plot of nodes' utilities, changing t_i vs. following the protocol. Each point represents a node. Points below the 45 degree line $y = x$ indicate that changing t_i yields lower utilities than following the protocol.

node's utility when it follows the protocol. We can see that all points are below the 45 degree line $y = x$, which means cheating always reduces a node's utility.

We further investigate the relationship between values of t_i and the received utility. In particular we observe the utilities of 4 selected nodes in one flow from node 4 to node 27. Each time we let one selected node change its t_i by $\pm 0.5t_i^\star$ or $\pm 0.8t_i^\star$, respectively, and other nodes remain cooperative. Fig. 5 shows the utilities per batch for different values of t_i. Clearly, for each node, the maximum utility per batch is achieved when $t_i = t_i^\star$, i.e., when the node follows the protocol.

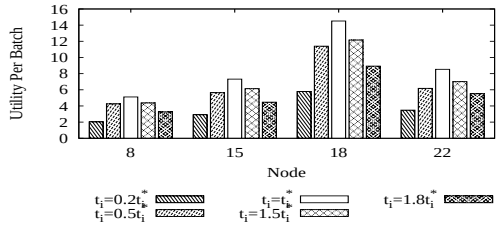

Fig. 5. Per batch utilities of four nodes when each of them uses different values of t_i. For each node, the utility is maximized when $t_i = t_i^\star$.

Underreporting Transmissions Or Improperly Punishing Downstream Nodes Now we consider the other two types of basic cheating behaviors. Fig 6 summarizes the results of the experiments on each of them, where the *utility gain* is defined as $\frac{\text{utility of cheating}}{\text{utility of following the protocol}} - 1$. In all the 100 runs, underreporting transmissions of upstream nodes always brings down the cheating node's own utility, by 4.65% to 44.03%. Similarly, in all the 100 runs, improperly punishing downstream nodes always leads to utility losses, ranging from 10.87% to 129.65%. Hence, neither type of cheating behavior can benefit the cheating node in any case.

C. System Convergence

When INPAC is used, the wireless mesh network has a stable state, namely the equilibrium state, in which all nodes faithfully follow the protocol. In this set of experiments, we study the procedure that the network system converges to the stable state.

At the beginning of each experiment, we let each node randomly select one of the following three behaviors: following the protocol, cheating by making only $0.5 * t_i^\star$ transmissions for forwarding each packet, and cheating by underreporting transmissions from an upstream node. After the experiment begins, each node repeatedly changes its behavior randomly.

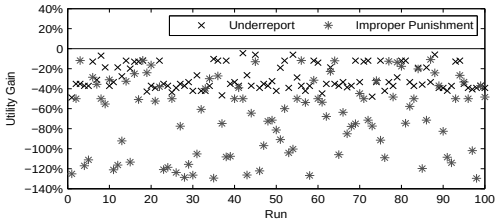

Fig. 6. Utility gains for underreporting transmissions of upstream nodes and for improperly punishing downstream nodes, respectively. Both are always negative. Neither type of cheating behavior can benefit the cheating node.

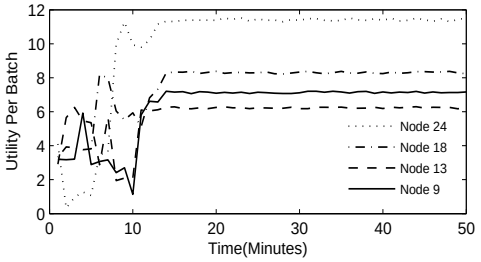

Fig. 7. Nodes' utilities when the network system converges to its stable state.

If the new behavior increases its own utility, the node moves to the new behavior; otherwise, it returns to its old behavior. The node terminates this procedure when it finds no way to further increase its own utility. When all nodes stop changing their behaviors, the entire network system is in a stable state.

We randomly pick 4 nodes in one experiment and observe their utilities in Fig. 7. (Due to limit of space, we cannot show the results of other experiments, which are similar.) As Fig. 7 shows, it takes about 10 minutes for the system to converge to a stable state, in which all nodes faithfully follow the protocol. Given the setup of our experiment, the convergence is fairly fast. We can make it even faster if the transactions are cleared more frequently.

D. Overheads

Now we evaluate the computation and communication overheads of INPAC. We measure the overheads in two different situations: when the system is in the stable state, and before the system converges to the stable state. Our experiments cover both the basic scheme and the extended scheme.

Overheads in Stable State We measure the overheads of both the INPAC basic scheme and the INPAC extended scheme, each in a session of transmitting 4800 packets, when the network system is in a stable state. In the extended scheme, we set $m = 6$. The results of our measurements are shown in Table I. We can see that the overheads of both the basic scheme and the extended scheme are reasonably low. If we compare the basic scheme with the extended scheme for the total overheads in the entire session, then the extended scheme is about 38.74% more efficient than the basic scheme, because the extended scheme has fewer operations of making reports.

Overheads in Convergence Before the system converges to a stable state, there may be additional overheads for punishing downstream nodes, which include the time for packet encryptions and decryptions. We use the 128-bit AES in ECB mode. On average, the overhead for punishing a downstream node is 0.143 ms per packet. We note that the keys for punishments

TABLE I
OVERHEADS OF INPAC.

Average time for processing a packet in either scheme	1.45 ms
Average time for making a report in either scheme	0.78 ms
Basic scheme's total overheads in entire session	4.75 s
Extended scheme's Total overheads in entire session	2.91 s

need to be set up in advance. In a network of size 30, the key setup time is 10.09 ms per node.

VIII. RELATED WORK

As we have mentioned in Section I, there are two types of existing work that are most related to this paper: incentive compatible routing in wireless networks using network coding, and incentive compatible packet forwarding in conventional wireless networks (not using network coding).

Incentive compatible routing in the context of network coding has been studied by Wu et al. [28]. They propose a method that stimulates wireless nodes' cooperation in the procedure of computing how many transmissions each node should make to forward each packet. In contrast, we start from the point when this routing decision has been made (i.e., when it has been decided how many transmissions each node should make to forward a packet), and propose an incentive scheme that stimulates nodes' cooperation in the procedure of actually forwarding packets. We note that Wu et al.'s work [28] and our work are complementary to each other, because the routing procedure and the packet forwarding procedure are closely related to each other and the proper functioning of the entire wireless network depends on both of them. We also note that their work cannot replace ours because the right number of transmissions being correctly computed does not necessarily mean the right number of transmissions will be made.

Incentive compatible packet forwarding has been extensively studied in the context of conventional wireless networks, which do not use network coding. The work in this category can be further divided into two subcategories: work using credit or virtual money (e.g., [5], [6], [12], [24], [30], among others), and work using reputation systems (e.g., [13], [19], among others). Our work also uses credit, but as we have explained in Section I, it is completely different from the existing solutions using credit, and cannot be replaced by any of them.

Our work is also related to the security studies of network coding [11], [29], because nodes may launch security attacks on the incentive scheme, and in order to prevent these attacks, we may borrow some ideas or techniques from existing security solutions. However, as we have emphasized, the main objective of our work is providing incentives rather than providing security. So our objective differs greatly from the objectives of security studies.

IX. CONCLUSION

In this paper, we propose INPAC, the first incentive scheme for packet forwarding in wireless mesh networks using network coding. It is complementary to the existing work on incentive compatible routing in the same type of wireless networks. Since packet forwarding is a fundamental procedure for computer networks, INPAC is of great importance to the application of network coding technology in environments with selfish users. We have extensively evaluated INPAC on the Orbit Lab testbed, and the results demonstrate that INPAC is both efficient and incentive compatible.

Since our scheme is designed to provide incentives to each individual node, one interesting open problem is to design a collusion resistant incentive scheme. We expect this open problem to be very challenging and plan to study it in our future work.

REFERENCES

[1] S. Akl and P. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans. Computer Systems*, 1(3):239–248, 1983.

[2] I. F. Akyildiz and X. Wang. A survey on wireless mesh networks. *IEEE Communications Magazine*, 43(9), 2005.

[3] L. Anderegg and S. Eidenbenz. Ad hoc-VCG: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *Proceedings of ACM MOBICOM*, 2003.

[4] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of ACM CCS 1993*, pages 62–73, 1993.

[5] L. Buttyan and J. P. Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *Proceedings of ACM MOBIHOC*, 2000.

[6] L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM Journal for Mobile Networks (MONET), special issue on Mobile Ad Hoc Networks*, summer 2002.

[7] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading structure for randomness in wireless opportunistic routing. In *Proceedings of ACM SIGCOMM*, 2007.

[8] S. Chen, P. Lin, D. Huang, and S. R. Yang. A study on distributed/centralized scheduling for wireless mesh network. In *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, pages 599–604, 2006.

[9] W. Dai. Crypto++5.5.2. Available at http://www.eskimo.com/ weidai/cryptlib.html.

[10] D. De Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of ACM MOBICOM*, 2003.

[11] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard. Resilient network coding in the presence of byzantine adversaries. In *Proceedings of IEEE INFOCOM*, 2007.

[12] M. Jakobsson, J. P. Hubaux, and L. Buttyan. A micropayment scheme encouraging collaboration in multi-hop cellular networks. In *Proceedings of Financial Crypto 2003*, La Guadeloupe, Jan. 2003.

[13] J. J. Jaramillo and R. Srikant. Darwin: Distributed and adaptive reputation mechanism for wireless ad-hoc networks. In *Proceedings of ACM MOBICOM*, 2007.

[14] S. Katti, S. Gollakota, and D. Katabi. Embracing wireless interference: Analog network coding. In *Proceedings of ACM SIGCOMM*, 2007.

[15] S. Katti, D. Katabi, H. Balakrishnan, and M. Medard. Symbol-level network coding for wireless mesh networks. In *Proceedings of ACM SIGCOMM*, 2008.

[16] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Medard, and J. Crowcroft. XORs in the air: Practical wireless network coding. In *Proceedings of ACM SIGCOMM '06*, Pisa, Italy, Sept. 2006.

[17] X.-Y. Li, Y. Wu, P. Xu, G. Chen, and M. Li. Hidden information and actions in multi-hop wireless ad hoc networks. In *Proceedings of ACM MOBIHOC*, 2008.

[18] http://madwifi.org.

[19] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of ACM MOBICOM*, 2000.

[20] Meraki Networks. http://meraki.com.

[21] MuniWireless LLC. http://www.muniwireless.com.

[22] M. J. Osborne and A. Rubenstein. *A Course in Game Theory*. The MIT Press, 1994.

[23] Rutgers ORBIT project team. http://www.orbit-lab.org.

[24] N. B. Salem, L. Buttyan, J. P. Hubaux, and M. Jakobsson. A charging and rewarding scheme for packet forwarding in multi-hop cellular networks. In *Proceedings of ACM MOBIHOC*, 2003.

[25] http://www.read.cs.ucla.edu/click/.

[26] W. Wan, X.-Y. Li, and Y. Wang. Truthful multicast in selfish wireless networks. In *Proceedings of ACM MOBICOM*, 2004.

[27] W. Wang, S. Eidenbez, Y. Wang, and X.-Y. Li. OURS-optimal unicast routing systems in non-cooperative wireless networks multihop routing in sensor networks. In *Proceedings of ACM MOBICOM*.

[28] F. Wu, T. Chen, S. Zhong, L. E. Li, and Y. R. Yang. Incentive compatible opportunistic routing for wireless networks. In *Proceedings of ACM MOBICOM*, 2008.

[29] Z. Yu, Y. Wei, B. Ramkumar, and Y. Guan. An efficient signature-based scheme for securing network coding against pollution attacks. In *Proceedings of IEEE INFOCOM*, 2008.

[30] 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 INFOCOM '03*, San Francisco, CA, Apr. 2003.

[31] S. Zhong, L. Li, Y. Liu, and Y. R. Yang. On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks — an integrated approach using game theoretical and cryptographic techniques. In *Proceedings of ACM MOBICOM*, 2005.

[32] S. Zhong and F. Wu. On designing collusion-resistant routing schemes for non-cooperative wireless ad hoc networks. In *Proceedings of ACM MOBICOM*, 2007.