CRISP: Collusion–Resistant Incentive–Compatible Routing and Forwarding in Opportunistic Networks

Umair Sadiq, Mohan Kumar, and Matthew Wright
University of Texas at Arlington
Arlington, Texas, USA
umair.sadiq@mavs.uta.edu, {mkumar,mwright}@uta.edu

ABSTRACT

In opportunistic environments, tasks such as content sharing and service execution among remote devices are facilitated by relays (devices with short–range wireless connectivity) that receive data, move around, and then forward the data. To achieve high throughput, it is important to secure forwarding and provide incentives for participation by relays. However, it is extremely challenging to monitor the behavior of relays in an opportunistic network due to sparse connectivity. Existing schemes do not work when selfish/malicious relays collude with each other to forge routing metrics, drop useful data, flood the network, or earn extra reward.

The credit scheme presented in this paper is the first in which routing as well as forwarding are incentive compatible. To design the scheme, the data transfer and loss in an opportunistic network are modeled as a non–linear generalized flow network. Then, optimality conditions for flow maximization describe the optimal behavior of a relay. This optimal behavior is made incentive compatible by requiring a relay to make a specific payment upon receiving the data and earn reward on forwarding the data. A cryptographic technique is used to make the scheme collusion resistant. Finally, a framework is proposed to implement CRISP in a completely distributed and opportunistic environment. Simulations on real and synthetic mobility traces validate a significant gain in throughput when compared with the existing credit schemes that are not incentive compatible.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design—*store and forward networks*

Keywords

Credit Scheme, Incentive Scheme, Opportunistic Networks

1. INTRODUCTION

The primary challenge in opportunistic networks is to send data in a reliable and efficient way despite the absence of a connected path between the end points. Data transmission is critical to every fundamental service that an opportunistic network can provide, including content sharing , dissemination, collection, and collaboration [7, 12, 14, 23]. The underlying mechanism used to forward data in these networks is to rely on any device in the area with short–range wireless connectivity to receive the data, expect the movements of the devices to make new connections possible, and to use these new connections to forward data onto other devices in the hope of eventually finding a path to the destination.

Numerous studies have investigated opportunistic routing in fully cooperative settings, in which each node participates in data forwarding to the best of its ability. Careful planning based on nodes' past and estimated future connection and movement behaviors can aid in making efficient forwarding in order to extract the most benefit out of each node's available resources [15, 19, 29]. In many realistic settings, however, we must anticipate selfish users who will not freely receive and send data at the cost of their devices' battery power and memory without some benefit to themselves. Additionally, we must be prepared to handle malicious users who seek to undermine the network's capabilities through manipulation of the routing system, e.g. by falsely claiming short routes to all destinations and dropping all data in a black hole attack, or by simply flooding the network.

Researchers have proposed a range of schemes to address these issues in opportunistic networks [17, 22, 37] as well as other network settings, such as ad hoc networks [31, 32, 35, 36]. The most promising approach to address these issues is to provide *credit* to nodes who participate in the forwarding process and allow only nodes who have credit to generate and send their own data packets[1]. Existing schemes, however, are not incentive compatible in routing or can be defeated by a small group of colluding attacker nodes [8, 17, 18, 20, 22, 24, 37]. Furthermore, existing schemes are designed to work in ad hoc networks in which the routing path is known [31, 35] or in mesh networks in which the routing and forwarding mechanisms are considered separately [10, 33]. To date, no incentive–compatible schemes are designed to operate in opportunistic environments in a fully distributed way without - i) an online trusted authority and ii) *a priori* knowledge of routes.

Contributions: In this paper, we present a new credit scheme called CRISP (Collusion–Resistant Incentive–ComPatible routing and forwarding) that addresses aforementioned challenges. The basic CRISP mechanism requires

[1]we use the term data, packets, and data packets interchangeably.

nodes to pay credit to receive a data packet and later obtain credit for forwarding that packet. Nodes must also pay credit to send new data packets of their own. Intuitively, selfish nodes who fail to forward packets will not be able to send anything, and malicious nodes are prevented from flooding the network. Further, routing black holes are greatly limited due to having to pay for receiving the data.

Although these mechanisms are intuitive, ensuring both collusion resistance and incentive compatibility for routing as well as forwarding is challenging. In this work, we describe a principled method for setting the payment parameters. In particular, we first convert a contact graph of the network into an instance of network flow problem and model the loss of packets in the network as a queueing theory problem with *reneging*. With this model, we find the optimal operating point of individual nodes to maximize overall network flow. We then design our credit–based incentive scheme so as to maximize the profit of a node when it operates at this optimal operating point. Our scheme thus makes it the selfish choice to receive and forward data packets in the most efficient manner for the network as a whole.

To enforce our scheme's requirements, we present a simple and secure protocol that makes CRISP resistant to colluding attackers. In particular, payments are secured using a Merkle hash tree [37] over the transaction history, and the histories are monitored by other nodes in the network. We also present a framework for maintaining and tracking the credits in a distributed manner.

CRISP is thus the first credit scheme that is appropriate for opportunistic networks. It is fully distributed, requires neither a central bank nor *a priori* knowledge of routing paths. CRISP is robust to flooding and blackhole attacks and provides incentives for nodes to participate fully in routing and forwarding operations.

Organization: We now layout the organization of the rest of the paper. In Section 2, we provide background on opportunistic networks and credit schemes so as to best motivate our design choices. Section 3 describes a model of the opportunistic network contact graph as a flow network and shows how we further model each node as a M/M/1 queue with reneging. Based on this model, Section 4 shows how to find the optimal arrival rate under a given function of loss (the *attenuation*). We then describe our credit scheme in Section 5, with payments and rewards set so as to make nodes obtain their maximum profit when accepting and forwarding data at the optimal arrival rate. Section 6 shows the results of analysis and simulations that demonstrate the effectiveness of our approach. Finally, we cover related work in Section 7 and conclude in Section 9.

2. DESIGN CONSIDERATIONS

In an opportunistic network, a node acts as a: *source* – generates data; *relay* – receives, stores and forwards data; and *destination* – receives data. Therefore, the possible cheating actions in each role can be: source – generates dummy data (flooding attack); relay – drops data (black–hole attack)/ randomly forwards data (to earn credit); or destination – does not acknowledge received data. In addition, nodes can also collude with each other for such attacks. Payment for generating data and a reward for acknowledging received data are used in [34] to secure against cheating by a source and destination. However, no existing credit schemes or security frameworks secure the behavior of a relay in face of collusion (we discuss further in Section 7). To this end, CRISP ensures truthful forwarding and honest reporting of routing utility even when nodes collude. We now describe the key concepts used to design CRISP.

Routing and Forwarding: In an opportunistic network, nodes come into contact with each other at different times and locations. Upon contact with another node, a data packet is forwarded, only if the encountered node is *more* likely to deliver the packet. This likelihood of delivery is computed by a routing/forwarding scheme that assigns every node, a utility value for each destination where a node with better likelihood of delivery has a higher value of utility. For example, bubble rap [15], maxprop [5], and others [19, 27, 29] assign utility values for social based forwarding, vehicle based routing, and general opportunistic networks respectively. Thus, let U_v^x represent the utility value of node v for destination node x that is computed by a corresponding routing scheme (e.g., [5, 15, 19, 27, 29]). A packet for destination node x is forwarded from node v to another node w if node w is more likely to deliver the packet, i.e., if $U_w^x > U_v^x$. A packet also has an expiration time such that if a packet is not delivered by that time, it is dropped/lost at the node. In this paper, we consider the problem of truthful *reporting* of utility, with the assumption that the node *knows* its true utility (e.g., a node only updates its utility based on contact with a (trusted) node with high reputation value).

Incentive Compatibility: Incentive compatibility means that a node earns highest reward when it behaves honestly. There are two related concepts: strong incentive compatibility means that honest behavior is the best strategy irrespective of strategy of other nodes, whereas, weak incentive compatibility means that honest behavior is the best strategy with the requirement that other nodes behave honestly. It was proved in [35] that forwarding cannot be incentive compatible in the strong sense[2]. Thus, we use the term incentive compatibility to mean weak incentive compatibility. Typically, credit schemes are incentive compatible in forwarding, i.e., a relay earns highest reward when it forwards the data to another relay or the destination itself. However, none of the existing credit schemes for opportunistic networks is incentive–compatible in routing.

The distinction between the forwarding and the routing stages was first highlighted in [35] for ad hoc networks. Techniques from mechanism design [21], are adapted in [35] to make the routing *and* the forwarding stage incentive–compatible. However, these techniques are not applicable in opportunistic networks because the complete path from source to destination is not known. A related problem is that of opportunistic routing in mesh networks, as the path is determined on a per–hop basis. In [33], routing is made incentive–compatible by a carefully designed incentive scheme. However, the forwarding stage is not considered. Then, in [10], forwarding in mesh networks is made incentive compatible with the *assumption* that routing is incentive compatible. Thus, a relay makes maximum profit by honestly behaving in forwarding *only if* it is assumed that the relay behaves honestly in the routing stage. When considered together, routing and forwarding are not incentive compatible because a node can make higher profit by cheating in

[2]strong incentive compatibility is equivalent to a mechanism in which honest behavior is the dominant strategy

both routing *and* forwarding. Therefore, it is important to consider routing and forwarding together, without any assumption about the honest behavior in two separate stages.

Collusion: In ad hoc networks, nodes declare a cost of transfer, and it has been shown in [32] that no mechanism can be incentive compatible when nodes collude in declaring the cost. In order to bypass this restriction, we fix the cost of forwarding, i.e., every node gets the same minimum reward upon forwarding. Since every relay that takes part in the forwarding gets a reward, colluding nodes add fake relays in the path. As a result, even those relays that do not take part in the forwarding get a reward. One recent work [9] addresses this issue by decreasing the amount of reward based on the number of relays in forwarding path. But this approach requires a large percentage of payment to the central credit authority (e.g. 57%,73% in case of 3, 4 relays). Thus far, it is not clear how to redistribute this over payment back in the nodes such that the original scheme remains incentive compatible (e.g., redistribution of extra payment among relays that contribute allows nodes to increase reward by adding fake relays). In addition, [9] does not secure the routing behavior of nodes i.e. a relay can increase its reward and decrease network throughput by modifying its routing utility to receive and randomly forward those packets.

Design Principles: CRISP is designed based on following principles:

1. Data is always forwarded to a relay with higher routing utility than the one at which data was received,
2. Reward is based upon successful delivery of data, and the amount of reward is proportional to the improvement in utility made by the relay.
3. The ratio of payment (for receiving data) to reward (upon forwarding data) is adjusted so that reporting incorrect routing utility results in reduced profit.

These principles are explained below.

Principle (1) is used to secure against random forwarding. To enforce this requirement, a cryptographic technique of layered coins [20, 37] is used. Upon exchanging a packet, relays in contact sign (using their private key) the routing utility at which the packet is received (see [20,37] for details). As a result, e.g., if a node v receives a packet for destination x when its utility U_v^x is 0.2, it can only forward it to another node w if $U_w^x > 0.2$ (because $U_v^x = 0.2$ is stored in the layered coin in an un-forgeable way). This prevents a node from randomly forwarding a packet.

Principle (2) is used to secure against collusions that add fake relays. Thus, if a node v receives a packet at utility U_v^x and forwards it to a node w with utility U_w^x, then, the reward is proportional to $U_w^x - U_v^x$. As a node forwards more packets, its expected reward also increases. However, adding the fake relays in the path does not increase the total profit because U_w^x and U_v^x stay the same (U_w^x, U_v^x are stored in the layered coin).

Principle (3) is used to secure against collusions to modify declared routing utility. Suppose a node declares a routing utility $\hat{U}_v^x$ instead of its true utility U_v^x. If $\hat{U}_v^x > U_v^x$, the node receives more packets but it becomes less likely to forward (find neighboring nodes with a $U_w^x > \hat{U}_v^x$) the packets. If $\hat{U}_v^x < U_v^x$, the node receives less packets, but is more likely to forward the packets. If $\hat{U}_v^x = U_v^x$, the node receives packets at a certain rate. This is called the optimal arrival rate as it maximizes the throughput (the number of packets that are delivered to destination) of the network. Note that a node is required to make a payment to receive a packet, therefore, a node always tries to forward the received packets to get a reward. For analysis, we formulate this scenario as a network flow problem.

Network Flow and Optimization: Two important extensions of network flow are: i) generalized flow – a flow that is not conserved across an arc and ii) dynamic flow – a flow that takes time to travel across an arc. An opportunistic network is better modeled by a generalized dynamic flow, with multiple commodities and concave gains. The multiple commodities are for packets routed to different destinations and the concave gains are to model the packets lost/expired before delivery to the destination. Closely related problem of generalized flow with concave gains has been studied in [28, 30] and more recently hardness results for generalized dynamic flow are proved in [13]. To keep our analysis tractable, we drop the requirement of dynamic flow and model the opportunistic network as a generalized flow with multiple commodities and concave gains, and validate our derived results through simulations.

3. SYSTEM MODEL

This section describes: (i) contact graph of opportunistic network; (ii) flow network model for data transfer between nodes; and (iii) queue model for data loss at nodes (data is dropped from the queue when not serviced in time).

3.1 Contact Graph

In an opportunistic network, nodes come into contact with each other at different times and locations. Consider the network operation in a time interval I of duration T (to the order of few minutes). We construct a directed contact graph $G = (V, E)$, where each vertex $v, w \in V$ represents a node in an opportunistic network with $n = |V|$, and each arc $vw \in E$ represents a contact between node v and node w at any point in time during the interval I. Every node $v \in V$ in a network acts as a source, relay and destination. Let d_{vw} represent the amount of data that can be sent from node v to node w during any number of contacts in the interval I. Note that d_{vw} is the duration of contacts multiplied by the bandwidth available for communication. We consider the network to be symmetric, i.e., if $vw \in E$ then $wv \in E$ and $d_{vw} = d_{wv}$. Note that $d_{vw} = 0$ if $vw \notin E$, and $d_{vv} := 0$.

3.2 Flow Network

Graph G can be modeled as a multi–commodity flow network G_g. In this network, the *amount of flow* is used to represent the *rate of data transfer*. Thus, the terms arrival, loss and forwarding rate of packets are equivalent to the incoming, lost and outgoing flow respectively. These terms are used interchangeably in rest of the paper.

Let c_{vw} represent the flow capacity between nodes v and w. Then $c_{vw} := d_{vw}/T$, i.e., the constant *rate* of data transfer possible between node v and node w for the entire interval I. Network G_g has multiple commodities of flows $x \in V$ which correspond to data en–route for different destination nodes $x \in V$. Let g_{vw}^x be a flow of commodity x. The amount of flow g_{vw}^x is used to represent the rate of transfer of data (destined to node x) from node v to node w. Data of a particular commodity is only transferred in one direction, i.e., if $g_{vw}^x > 0$ then $g_{wv}^x = 0$. Let b_v^x be the amount of flow of commodity x generated at node v. Flow b_v^x is used to represent rate for the data generated during the interval I

or available at the start of interval I at node v. All flows are non–negative and the capacity constraint requires that the sum of flows of all commodities between nodes v and w is less than or equal to the capacity ($c_{vw} = c_{wv}$), i.e.,

$$\sum_{x \in V} g_{vw}^x + \sum_{x \in V} g_{wv}^x \leq c_{vw} \quad \forall v, w \in V, \tag{1}$$

Data packets have an expiration time, and a packet is lost at a node if it is not delivered before the expiration time. Let Q_v^x be the probability that a packet of commodity x is lost at node v. Then, if a node v receives packets for destination node x at the rate $\lambda_v^x = \sum_{w \in V} g_{wv}^x$, it is able to forward at the rate $\Gamma_v^x(\lambda_v^x) = \lambda_v^x(1 - Q_v^x)$.

Note that the data is lost at the node itself. This can be modeled in terms of a generalized flow network by splitting each vertex into two vertices joined by a new arc of flow λ_v^x, and then adding the attenuation function $\Gamma_v^x(\lambda_v^x)$ over the flow of this new arc. This process is similar to the conversion of a flow network with vertex capacities to a flow network with arc capacities. However, for notational simplicity, we avoid the standard notation, and describe the flow conservation as follows. For each node and commodity, the sum of outgoing flows is equal to the sum of locally generated flow and the attenuated incoming flow, i.e., for all $v, x \in V \quad \sum_{w \in V} g_{vw}^x = \Gamma_v^x(\lambda_v^x) + b_v^x$.

Notation: Let $\lambda_\mathbf{v} = (\lambda_v^x, \forall x \in V - \{v\})$ represent the column vector of arrival rate (incoming flow) of all commodities (except commodity destined to node v) at node v, such that, $\lambda_\mathbf{v} \in \mathbb{R}^{(n-1)\times 1}$. Let $\lambda = (\lambda_\mathbf{v}, \forall v \in V)$ represent the column vector of arrival rate (incoming flow) of all nodes and commodities (except commodities destined to each node), such that, $\lambda \in \mathbb{R}^{n(n-1)\times 1}$. Let $\hat{\lambda}_\mathbf{v}$ and $\hat{\lambda}$ represent the corresponding optimal arrival rate vectors. Let $\mathbf{h_v} = \lambda_\mathbf{v} - \hat{\lambda}_\mathbf{v}$ and $\mathbf{h} = \lambda - \hat{\lambda}$ represent the difference between a general and optimal rate vector where $h_v^x = \lambda_v^x - \hat{\lambda}_v^x$ is used to represent an individual component. Vector $\mathbf{S_v}$ and $\mathbf{S}$ are used to show sum of vector components, where $\mathbf{S_v} = (1, \forall x \in V - \{v\})$ and $\mathbf{S} = (1, \forall v \in V, x \in V - \{v\})$ are row vectors of ones. For example, $\mathbf{S_v}\lambda_\mathbf{v} = \sum_{x \in V - \{v\}} \lambda_v^x$ is the total incoming flow (excluding commodity v) at node v, similarly, $\mathbf{S}\lambda = \sum_{v \in V} \sum_{x \in V - \{v\}} \lambda_v^x$ is the total incoming flow at all nodes (excluding commodities destined to each node). The total outgoing flow (excluding locally generated flow) at all nodes is represented by $\Gamma(\lambda) := \sum_{v \in V} \Gamma_v(\lambda_\mathbf{v})$ where $\Gamma_v(\lambda_\mathbf{v}) := \sum_{x \in V - \{v\}} \Gamma_v^x(\lambda_v^x)$ is the outgoing flow at node v.

3.3 Queue Model

Packets received by a node have some expiration time, therefore, not all packets are forwarded. If a node v receives commodity x at rate λ_v^x, then, the rate *available* to forward the received commodity is μ_v^x, such that

$$\lambda_v^x + \mu_v^x = c_v^x. \tag{2}$$

The net capacity of relaying commodity x through node v is limited by capacity c_v^x, where c_v^x depends on the capacity c_{vw} available with each node w, the rate b_v^x consumed by forwarding the data generated at node v, and the schedule for splitting available capacity c_{vw} for each commodity x. In our solution, we look at the properties of the optimal arrival rate λ_v^x for a given capacity c_v^x for each node and commodity.

The rate available for forwarding μ_v^x is different from rate at which commodity x is actually forwarded $\Gamma_v^x(\lambda_v^x)$. For example, a node can receive data at the rate (in packets/min)

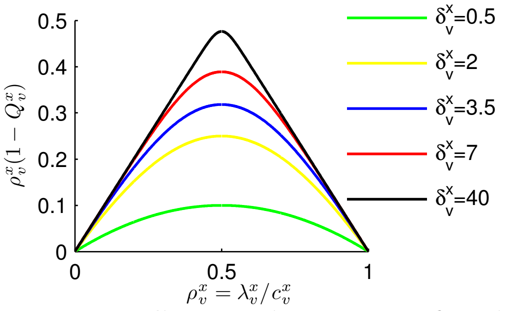

Figure 1: Forwarding rate is a concave function of packet arrival rate.

$\lambda_v^x = 5$ with the available rate of forwarding $\mu_v^x = 10$, but only able to forward at $\Gamma_v^x(\lambda_v^x) = 3$. $\Gamma_v^x(\lambda_v^x) \leq \lambda_v^x$ because a node cannot forward more packets than it receives, and some packets expire when there is no contact opportunity to forward the packet.

To derive the packet loss rate, as an approximation, the arrival rate λ_v^x and available forwarding rate μ_v^x are modeled as a Poisson Process. This scenario represents a $M/M/1$ queue with reneging [3]. The arrival process is Poisson with rate λ_v^x, the service process corresponds to forwarding process which is Poisson with rate μ_v^x, and each node v for a flow commodity x is a server. The holding time of a packet in queue is the time after which a packet is dropped if its service has not been completed i.e. if the packet has not been forwarded. We assume a constant holding time δ_v^x / c_v^x, where δ_v^x is a dimensionless constant that gives the order of holding time with respect to capacity of a node c_v^x.

4. ANALYSIS

The value of utility that a node declares determines the arrival rate of packets and consequently the packet loss rate. In this section we derive the relation for packet loss, and use it to find the optimal packet arrival rates that maximize the network flow. We call it the optimal packet arrival rate, as it corresponds to the optimal value of utility that a node should declare to maximize the network throughput.

4.1 Attenuation due to expired packets

We use the result (equation (29) in [3]) for the probability of a dropped packet Q_v^x in a $M/M/1$ queue with reneging. Substituting μ_v^x from (2) and replacing λ_v^x / c_v^x with ρ_v^x, the probability of packet loss is given by

$$Q_v^x = \frac{(1 - 2\rho_v^x)e^{\delta_v^x(2\rho_v^x - 1)}}{1 - \rho_v^x(1 - e^{\delta_v^x(2\rho_v^x - 1)})} \qquad \text{when} \quad \rho_v^x \neq 0.5,$$

where, $Q_v^x = 2/(2 + \delta_v^x)$ when $\rho_v^x = 0.5$. Though we were unable to prove, but from simulations we find that for a fixed value of δ_v^x, the function $\rho_v^x(1 - Q_v^x)$ is concave (first derivative decreases monotonically) in terms of ρ_v^x. Figure 1 shows the function $\rho_v^x(1 - Q_v^x)$ for different values of δ_v^x. Using a known property, that perspective of a concave function is also concave (replacing ρ_v^x by λ_v^x / c_v^x and multiplying by c_v^x) [4], we find that $\Gamma_v^x(\lambda_v^x) = \lambda_v^x(1 - Q_v^x)$ is concave in terms of λ_v^x.

4.2 Optimal flow with concave attenuation

In order to maximize the total flow delivered to the destination nodes, we minimize the total flow lost at individual relays given that all generated flow is pushed into the network. Using notation defined in Section 3.2, the total flow lost at all nodes is the sum of total flow lost at individual nodes for all commodities and is given by, $\mathbf{S}\lambda - \Gamma(\lambda)$.

Let the optimal incoming flow (arrival rate) at individual nodes be $\hat{\lambda}$ and the sum of optimal incoming flows be $L := \mathbf{S}\hat{\lambda}$. In order to find the properties of the optimal incoming flow that maximize the delivered flow, our objective is to minimize $L - \Gamma(\lambda)$ (total flow lost at relays) which is equivalent to maximizing $\Gamma(\lambda)$ such that $\mathbf{S}\lambda = L$. Also, flows cannot be negative, i.e., we have $\lambda \geq 0$, where $\geq$ is used for component wise inequality. Formally, this is described as,

$$\begin{aligned} \text{maximize} \quad & \Gamma(\lambda) \\ \text{subject to} \quad & \mathbf{S}\lambda = L, & (3) \\ & \lambda \geq 0. & (4) \end{aligned}$$

There is one additional capacity constraint on λ that we do not consider here. This is described in the form of individual flows g_{vw}^x in (1). One can solve the above optimization problem for the exact feasibility constraints. This problem is similar in structure to the problem of network utility maximization [16]. The concave attenuation function $\Gamma_v^x(\cdot)$ points to the law of diminishing returns, i.e., probability of successfully forwarding a packet decreases as the node declares a high utility value in routing to receive more packets. However, we relax the flow capacity constraint (1) to constraint (3) in order to analyze the optimal rate $\hat{\lambda}$. The reason is that, specifically in opportunistic network, the topology is not static over time, and as nodes move around the rate $\hat{\lambda}_v^x$ of one node (commodity) can be shifted to another node (commodity). In addition, simulation in a range of environments shows that honest behavior of nodes in routing and forwarding yields arrival rates that are very close to the optimal rate $\hat{\lambda}$ with constraints (3) and (4) only.

Based on analysis of attenuation in Section 4.1, assume that $\Gamma_v^x(\lambda_v^x)$ is concave and differentiable. Then, $-\Gamma_v^x(\lambda_v^x)$ is convex and $-\Gamma(\lambda)$ is convex because it is an increasing convex function of a convex function. Also, the constraints (3) and (4) are affine and convex. Solving the *Lagranian*, the optimal point is the global maximum and Karush–Kuhn–Tucker conditions for optimality [4] give us

$$\nabla(-\Gamma(\hat{\lambda})) + \xi\nabla(\mathbf{S}\hat{\lambda} - L) + \sum_{v\in V}\sum_{x\in V-\{v\}} \xi_v^x \nabla(-\hat{\lambda}_v^x) = 0,$$

$$\forall v \in V, x \in V - \{v\} \quad \frac{\partial \Gamma_v^x(\hat{\lambda}_v^x)}{\partial \lambda_v^x} = \xi - \xi_v^x, \quad (5)$$

$$\forall v \in V, x \in V - \{v\} \qquad \xi_v^x\ \hat{\lambda}_v^x = 0, \qquad (6)$$

where the Lagrange multipliers $\xi, \xi_v^x \geq 0$. Here $\xi \geq 0$ due to optimal value of L. It can be shown that $\Gamma(\lambda)$ for $\xi = 0$ is always greater than the case when $\xi < 0$ by changing constraint (3) to be $\mathbf{S}\lambda \leq L$. Equation (5) is the stationary condition, whereas (6) is the complementary slackness condition. These are in addition to the primal feasibility constraints (3) and (4). These optimality conditions can be further simplified. From (6), if $\hat{\lambda}_v^x > 0$, then $\xi_v^x = 0$ and $\partial\Gamma_v^x(\hat{\lambda}_v^x)/\partial\lambda_v^x = \xi$. So, our final result summarizes the optimality conditions of (3), (4), (5), and (6) as follows.

Main Result: Let $\hat{\lambda}$ be the arrival rate that satisfies primal feasibility constraints (3) and (4). Suppose there exists a constant $\xi \geq 0$ such that $\forall \hat{\lambda}_v^x > 0$,

$$\frac{\partial \Gamma_v^x(\hat{\lambda}_v^x)}{\partial \lambda_v^x} = \xi, \qquad (7)$$

then $\hat{\lambda}$ is the optimal arrival rate, i.e., it maximizes $\Gamma(\hat{\lambda})$.

4.3 Optimal flow with general attenuation

In the previous section we derived the optimal arrival rate for continuous, differentiable and concave attenuation function. Our result on concavity of attenuation function is from an approximate $M/M/1$ queue model. Therefore, in this section, we consider general function $\Gamma_v^x : \mathbb{R}_+ \to \mathbb{R}_+$, without any requirement of continuity, differentiability and concavity. Following derivations use the insight about optimality conditions from the case of continuous concave function.

Theorem 1. *Let $\hat{\lambda}$ be the arrival rate that satisfies feasibility constraints (3) and (4). Suppose there exists a constant $\xi \geq 0$ such that $\forall \hat{\lambda}_v^x > 0$,*

$$\frac{\Gamma_v^x(\lambda_v^x) - \Gamma_v^x(\hat{\lambda}_v^x)}{\lambda_v^x - \hat{\lambda}_v^x} \geq \xi, \quad \forall \lambda_v^x \in [0, \hat{\lambda}_v^x), \text{ and}$$

$$\frac{\Gamma_v^x(\lambda_v^x) - \Gamma_v^x(\hat{\lambda}_v^x)}{\lambda_v^x - \hat{\lambda}_v^x} < \xi, \quad \forall \lambda_v^x > \hat{\lambda}_v^x,$$

then $\hat{\lambda}$ is the optimal arrival rate, i.e., it maximizes $\Gamma(\hat{\lambda})$.

Proof. In order to prove optimality, we show that $\Gamma(\hat{\lambda}) \geq \Gamma(\lambda)$ for any λ that satisfies the constraints (3) and (4). We can express any general rate $\lambda_v^x \geq 0$ in terms of the optimal $\hat{\lambda}_v^x$ as follows $\lambda_v^x = \hat{\lambda}_v^x + h_v^x$. Then, the conditions in the theorem can be re–written as

$$\Gamma_v^x(\hat{\lambda}_v^x) - \Gamma_v^x(\hat{\lambda}_v^x + h_v^x) \geq -\xi h_v^x, \quad \forall h_v^x \in [-\hat{\lambda}_v^x, 0),$$

$$\Gamma_v^x(\hat{\lambda}_v^x) - \Gamma_v^x(\hat{\lambda}_v^x + h_v^x) > -\xi h_v^x, \quad \forall h_v^x > 0,$$

Summing the above two relations over all nodes and commodities, we get

$$\Gamma(\hat{\lambda}) - \Gamma(\lambda) \geq -\xi \mathbf{S}\mathbf{h}. \qquad (8)$$

From constraint (3), $\mathbf{S}\lambda = \mathbf{S}\hat{\lambda}$. Substituting $\lambda = \hat{\lambda} + \mathbf{h}$, $\Rightarrow \mathbf{S}(\hat{\lambda} + h) = \mathbf{S}\hat{\lambda} \Rightarrow \mathbf{S}\mathbf{h} = 0$. Substituting this result back in (8), we get $\Gamma(\hat{\lambda}) - \Gamma(\lambda) \geq 0 \Rightarrow \Gamma(\hat{\lambda}) \geq \Gamma(\lambda)$. □

Properties of Optimal Flow: The intuitive idea is simple. If *additional* packets are to be received at rate 10, the relay responsible for forwarding should be the one that can forward at rate 7 instead of the one that can only forward at rate 6. Therefore, with optimal allocation of flows, shifting individual flows from one node to another does not increase the overall forwarding rate. To see this property, let $M_v^x(\phi) = \frac{\Gamma_v^x(\lambda_v^x+\phi) - \Gamma_v^x(\lambda_v^x)}{\phi}$. Then $\phi M_v^x(\phi)$ is the increase in forwarding rate for increasing *optimal* arrival rate by ϕ. Similarly $\phi M_w^x(-\phi)$ is the decrease in forwarding rate for decreasing *optimal* arrival rate by ϕ. The optimal allocation of flows requires that the increase in forwarding rate $\phi M_v^x(\phi)$ at one node is always less than the decrease in forwarding rate $\phi M_w^x(-\phi)$ at another node, for the same amount of change ϕ in arrival rate, i.e., $\phi M_w^x(-\phi) > \phi M_v^x(\phi) \Rightarrow M_w^x(-\phi) > M_v^x(\phi)$. This condition is true because Theorem 1 requires $M_w^x(-\phi) \geq \xi$ and $M_v^x(\phi) < \xi$ at all values of $\phi \in (0, \hat{\lambda}_w^x]$ and $\forall v \in V, x \in V - v$. Also, the ratio of forwarding rate to the arrival rate is always greater than or equal to ξ (substituting $h_v^x = -\hat{\lambda}_v^x$ in Theorem 1 gives $\Gamma_v^x(\hat{\lambda}_v^x)/\hat{\lambda}_v^x \geq \xi$).

Consider the arrival (and corresponding forwarding) rates of a node in Table 1. Based on Theorem 1, for any value of $\xi \in (0.05, 0.85)$, the optimal arrival rate is $\lambda_v^x = 70$ with forwarding ratio 0.86. In a traditional credit scheme that rewards nodes based on the number of packets forwarded, a node maximizes its profit by maximizing its forwarding rate ($\Gamma_x^v = 62$) at a much lower forwarding ratio of 0.56. This is the key difference between CRISP and existing schemes and results in significant improvement in network throughput.

Table 1: CRISP operates at forwarding ratio 0.86 for any value of $\xi \in (0.05, 0.85)$.

Forwarding ratio	0.9	**0.86**	0.61	0.56	0.50
Forwarding rate Γ_v^x	9	**60**	61	62	60
Arrival rate λ_v^x	10	**70**	100	110	120

We call ξ the threshold constant and evaluate its effects further in Section 6.

5. CREDIT SCHEME, CRISP

In the previous section, we derived the optimal rate at which nodes should receive and forward the data. In this section, we design the credit scheme in which receiving and forwarding at the exact same rate maximizes a node's profit.

Payments and Rewards: We require a node to make a payment $\beta > 0$ to receive a data packet and earn a reward $\alpha > \beta$ for forwarding a data packet. The rate of profit P_v^x a node v makes on receiving commodity x at rate λ_v^x and forwarding at rate $\Gamma_v^x(\lambda_v^x)$ is given by $P_v^x = \alpha\Gamma_v^x(\lambda_v^x) - \beta\lambda_v^x$. Using notation from Section 3.2, the net rate of profit P_v at node v is given by $P_v = \sum_{x \in V - \{x\}} P_v^x = \alpha\Gamma_v(\lambda_\mathbf{v}) - \beta\mathbf{S_v}\lambda_\mathbf{v}$.

5.1 Incentive Compatibility

Based on the above payment mechanism, we show that setting the correct ratio of payment to the reward results in maximizing the profit at the optimal rate of arrival and hence, the credit scheme is incentive compatible.

THEOREM 2. *Suppose a node receives data at rate $\mathbf{S_v}\lambda_\mathbf{v}$, and forwards at rate $\Gamma_v(\lambda_\mathbf{v})$. The profit $P_v = \alpha\Gamma_v(\lambda_\mathbf{v}) - \beta\mathbf{S_v}\lambda_\mathbf{v}$ is maximum at the optimal rate of arrival $\hat{\lambda}_\mathbf{v}$ when β/α is set equal to ξ.*

PROOF. Let $\hat{\lambda}_\mathbf{v}$ be the vector of optimal rate of arrival as defined in Theorem 1, and the corresponding profit be $\hat{P}_v$. Next we show that $\hat{P}_v \geq P_v$. From the definition of profit, $\hat{P}_v - P_v = \alpha(\Gamma_v(\hat{\lambda}_\mathbf{v}) - \Gamma_v(\lambda_\mathbf{v})) - \beta\mathbf{S_v}(\hat{\lambda}_\mathbf{v} - \lambda_\mathbf{v})$. Using steps similar to the ones used to derive (8), we get

$$\hat{P}_v - P_v \geq -\alpha(\xi\mathbf{S_v}\mathbf{h_v}) - \beta\mathbf{S_v}(\hat{\lambda}_\mathbf{v} - \lambda_\mathbf{v})$$
$$\Rightarrow \hat{P}_v - P_v \geq (-\alpha\xi + \beta)\mathbf{S_v}\mathbf{h_v}$$

When $\beta/\alpha = \xi$, we get $\hat{P}_v - P_v \geq 0 \Rightarrow \hat{P}_v \geq P_v$. $\square$

Credit Scheme: For $\beta > 0$ and $\alpha := \beta/\xi$, the exact payments to network and rewards from network for a node's role as source, relay and destination are as follows.
Relay: The payment to receive a packet is β and (expected) reward upon forwarding a packet is $\alpha := \beta/\xi$.
Destination: The payment to receive a packet is β and reward upon acknowledging the received packet is β.
Source: The payment to generate a packet is a fixed amount to cover net cost on network.

5.2 Collusion Resistance

Source, relays and destination can collude with each other to increase their profit, e.g., a source can give destination a behind–the–scene compensation to not acknowledge the received packet (as a result the source will not have to pay for the reward of relays). Possibilities of such collusions are analyzed in [34] which provides a number of conditions to ensure collusion resistance. In CRISP, the reward of relays when the packet is not delivered is zero (corresponds to $\gamma = 0$ in [34]). Based on analysis in [34], the only additional requirement to make CRISP collusion-resistant to such manipulations is that the source makes the same fixed payment to the network irrespective of whether the packet is delivered or not. When the packet is delivered, each node gets a reward based on its improvement, otherwise the payment stays in the network. However, analysis in [34] does not consider collusion among relays to add fake relays or to change routing utility. CRISP secures against these collusions [Section 2] as adding fake relays does not increase profit, and changing routing utility reduces profit.

5.3 Opportunistic Framework

Each relay pays β to the previous relay in the path in order to receive a packet. If the packet is delivered, the source makes the final payment to the relays and the destination; otherwise, the source pays to the network. To enforce this above payment scheme, any of the existing frameworks [20, 37] can be used. Here, we briefly discuss a completely decentralized framework, without any requirement of an offline bank. Relaxing the requirement of an offline bank is useful for maintaining credits on sensor nodes and devices that do not connect to a central server often enough to validate earned credits. Existing schemes have an online or an offline bank, which is a trusted authority. Each node submits all the transaction receipts or a summarized report of all transactions to the bank which can then check if a node is cheating and revoke its credits or its certificate. The problem becomes very challenging when such a trusted bank is not available to check all the receipts. A trivial solution is to keep track of all transaction receipts on each and every node, but it is prohibitive due a large overhead of maintaining these receipts on every node.

Forged Credits: To this end, we propose a novel solution based on a simple idea – *credit is neither created nor destroyed*; it is only transferred between nodes. Thus, each new node joining the network is initialized at some value, and then the only way for it to earn credit is by relaying traffic of other nodes. Thus, in the long run, a node provides the network some service proportional to the resources it consumes for itself. To identify collusion between two nodes to forge credit, each node also keeps track of the credit value at all other nodes. Then, if the value of total credit in the network does not stay constant, nodes can track back as to which node tried to reuse the credit.

Ensuring Payments: In case of a delivered packet, a source *has* to make a payment, because relays can identify the source of the packet, and upon not receiving the payment they can discontinue providing the service for non–paying source. To ensure a node's payment to the network when the packets are not delivered, a Merkel tree is used. We do not provide details here as a similar use for maintaining offline credit is proposed in [20, 37]. The difference is the leaf value of tree – in our case it is the hash of the packet a source generates. The root value of tree is flooded in the network by each source. To ensure that a source makes the payment, relays request the support of hash tree for all the packets that the relay forwarded. The overhead is just O(1) stored value per relay and verification of $O(\log(m))$ constant–size (e.g. 160–bit) hashes for m variable–sized packets that a relay forwards.

Consistency: To maintain and update credits of other nodes in a consistent manner, temporal distance [27, 29] is used. To secure the credits, each node signs the time–

stamped credit value with its private key. When two nodes come into contact, the node with smaller temporal distance with the destination, updates the other. Also, payments can be made either in a distributed manner through other nodes, or upon a direct contact with a relay. Payment upon a direct contact takes longer, but has low overhead in comparison to the other mechanism. In the next section, we evaluate our credit scheme. Due to limited space, we leave the detailed description and evaluation of payment framework for future work.

6. EVALUATION

A relay pays β credit to receive a packet and earns α credit upon forwarding a packet under the CRISP scheme. It has been shown in several studies that cheating in the routing and forwarding stage results in poor network performance [11]. Therefore, we consider a special type of cheating that is possible in the existing credit schemes. Overall, we consider three types of behavior:

- **Honest:** The relay truthfully follows the routing and forwarding protocol.
- **Selfish:** The relay modifies the routing utility to operate at the optimal rate [Theorem 1].
- **Cheating:** The relay modifies the routing utility to maximize the forwarding rate.

Existing schemes promote cheating (a node maximizes its profit by maximizing the forwarding rate) and therefore suffer from reduced network throughput. In our evaluation, we show that,

1. Cheating reduces the network throughput compared to honest behavior.
2. CRISP discourages cheating and promotes honest and selfish behavior.
3. Selfish behavior result in similar throughput compared to honest behavior.

CRISP promotes honest behavior and beneficial selfish behavior (by taking into account the packets received in addition to the packets forwarded) and thereby maintains good throughput. We show all of the above properties hold for a wide range of ξ values, packet expiration times, and packet generation rates in both real and synthetic mobility environments.

Simulation setup: Extensive simulations are run on real and synthetic mobility traces. The real trace was collected from people walking in a state fair [26], and synthetic trace is generated from a levy walk mobility model [25]. Messages are generated at each node for a random destination. The real mobility trace has been collected from participants that carry GPS receivers which log position at 30 second intervals. In order to make a comparison with suitable number of nodes, track logs from each day are considered to be a separate user. These logs have durations from around one to ten hours each day. We truncate all logs to 90 minutes during which 18 users record their location. Traces with levy walk are generated with the settings recommended in [25] with an area of $500 \times 500m^2$. All plots show the average of five simulation runs and the confidence interval is plotted on the first standard deviation. We use temporal distance [27, 29] as the routing scheme in simulation. All values of profit are normalized by the profit of selfish node and the value of throughput is normalized by the value of honest node.

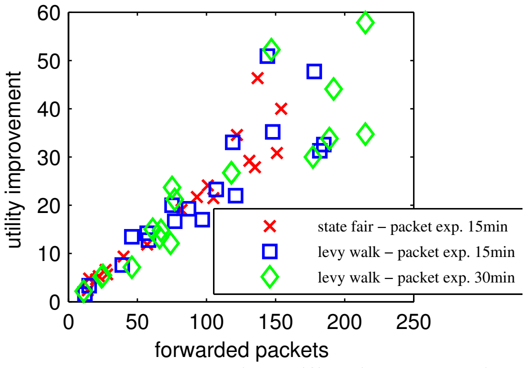

Figure 2: Improvement in utility is proportional to number of forwarded packets

6.1 Collusion Resistance

To secure against collusion to add fake relays, profit should be based on the total improvement a node makes in the path towards the destination as described in Section 2. Figure 2 shows that the expected reward based on improvement in utility is linear in the number of packets that a node forwards in different sets of environments. Thus, a reward based on improvement is equivalent to a reward based on number of packets (as described in our analysis) that a node forwards. Here, the value utility is an approximation for improvement that needs to be made to deliver the packet. We study the routing metric of temporal distance as is used in [19, 27, 29] and compute the utility by extending a relation derived in [27]. Thus, for a temporal distance t, the utility is computed as e^{-kt}, where k depends on the diffusion of the node (we take a value of 0.4/min, e.g., a 0.36 utility at a temporal distance of 2.5 mins). Thus, a node that receives a packet at $t = 10$ min and forwards it to a node at $t = 8$ min; receives an award of $e^{-0.32} - e^{-4}$ multiplied by the appropriate constant that makes the expected reward equal to α.

6.2 Value of threshold constant ξ

In order to find the actual value of ξ that maximizes throughput, we use the queue approximation result as described in Section 4.1. Let ψ be the average forwarding ratio among all nodes, i.e. the ratio of total packets forwarded to the total packets received over the whole network. From (7), the value of ξ is equal to $\frac{\partial \Gamma_v^x(\lambda_v^{x\star})}{\partial \lambda_v^x}$ where $\lambda_v^{x\star}$ is the solution for equation $\frac{\Gamma_v^x(\lambda_v^{x\star})}{\lambda_v^{x\star}} = \psi$. We only need to know value of δ_v^x in order to solve this equation[3]. In an actual network, the forwarding ratio of individual nodes is much different from the average value ψ. However, the arrival rates of all individual nodes still follow conditions of Theorem 1 for a constant value of ξ computed using the average value of ψ.

From simulation, we find ψ and the value of ξ that maximizes the throughput. Figure 3 compares the results from simulation and analysis. The lines show the mapping of forwarding ratio ψ to theoretically computed values of ξ at three different values of δ_v^x. The scatter points show the actual value ξ that maximizes throughput at the corresponding forwarding ratio ψ. Changing the generation rate or the

[3]The value of c_v^x is not required because ξ effectively depends on the ratio $\lambda_v^x / c_v^x = \rho_v^x$, i.e., $\xi = \frac{\partial(\rho_v^{x\star}(1-Q_v^x))}{\partial \rho_v^x}$ where $\rho_v^{x\star}$ is solution of equation $\frac{\rho_v^{x\star}(1-Q_v^x)}{\rho_v^{x\star}} = \psi$.

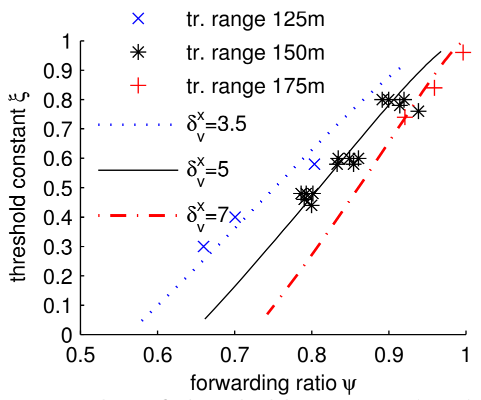

Figure 3: Value of threshold constant ξ – simulation and analysis

packet expiration time changes the forwarding ratio ψ at nodes. In a specific environment, the value of ξ with different forwarding ratios (generation rate or the packet expiration time) lies close to the line corresponding to a fixed value of δ_v^x. Changing the environment from being sparse (transmission range 125m, forwarding ratio 65%) to more connected (transmission range 175m, forwarding ratio 98%) corresponds to the value of δ_v^x from 3.5 to 5. Thus, at a given forwarding ratio ψ, using the values of $\delta_v^x \in [3.5, 5]$ one can estimate a range such that ξ that maximizes throughput belongs to that range. Later we show that CRISP is robust against inaccurate estimate of ξ.

6.3 Performance across nodes

In Fig. 4 and Fig. 5, each point corresponding to a node id is a complete simulation run in which that node's behavior is honest, selfish or cheating. Figure 4 shows that the network throughput when a node is honest or selfish is very close, but drops sharply when a node cheats. For example, the network throughput decreases by 15% when node #11 is cheating compared to the network throughput when node #11 is honest or selfish. The throughput degradation depends highly on the location and contacts of the node. For this reason, we show separate results for each of the cheating nodes. When two or more nodes cheat, then depending on their individual impact, they can drop a large percentage of data. For example, if node #1 and #2 cheat, they may only decrease the throughput by 5%, whereas if node #9 and #11 cheat, they can drop up to 25% of total data as can be seen in Fig. 4.

Figure 5 shows that the profit under CRISP of a selfish node is highest, a honest node is very close to it, and a cheating node is much lower (mostly negative). For example, when node #11 cheats, its profit is 500% less than when it is honest or selfish. The exception is when node #5 or #15 cheat, they make more profit than honest, but still less than selfish. This is not a problem because the throughput of selfish behavior is very close (same for node #5, slightly lower for node #15) to that of honest behavior as shown in Fig. 4.

6.4 Robustness and incentive compatibility

The value of ξ that maximizes throughput is 0.65 in Fig. 6 and 7 for the state fair trace. These figures show the results averaged over all nodes such that in one simulation run only a single node changes its behavior (honest, selfish or cheating). Figure 7 shows that profit under CRISP of honest (and selfish) behavior is greater than cheating for all

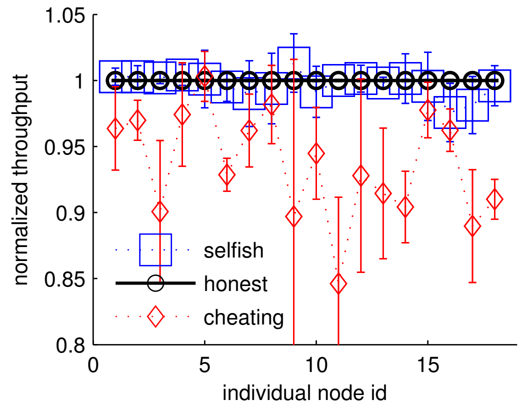

Figure 4: A single cheating node decreases network throughput by up to 15% (node #11).

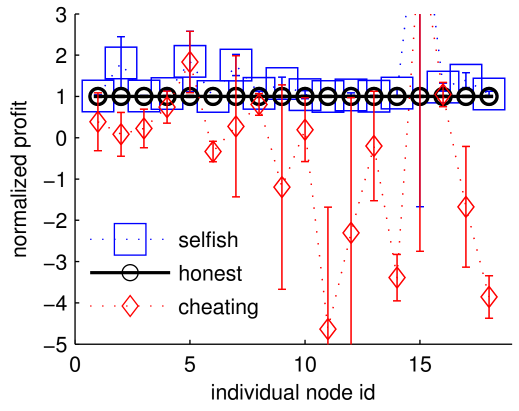

Figure 5: CRISP reduces the profit of cheating by up to 500% (node #11)

values of threshold constant $\xi > 0.4$. Also, for all values of ξ, Fig. 6 shows that throughput of selfish (and honest) behavior is greater than cheating. In real time, a node cannot figure out the exact utility it should declare for selfish or cheating behavior. In our simulations, we try out different (constant) shifts in the routing utility and plot the results only for the change in routing utility that maximizes forwarding rate (cheating) or profit under CRISP (selfish). Thus, CRISP is quite robust to inaccurate values of ξ and maintains incentive compatibility for a range (0.5 to 0.7) of ξ values because a node is guaranteed a good profit just by honestly routing and forwarding.

6.5 Performance across packet expiration times and packet generation rates

Figures 8 and 9, show mean throughput when packet expiration time and packet generation rates are varied in the state fair mobility trace. Each point corresponds to results averaged over 90 simulation runs (5 for each of 18 individual nodes when that node is honest, selfish or cheating). It is evident from Fig. 8 and 9 that the network throughput with selfish nodes is within 1% of that of the honest nodes whereas that of cheating nodes is around 5% lower. Note that the throughput is averaged over all nodes such that in one simulation only a single node changes its behavior (honest, selfish or cheating). Thus, the impact of multiple cheating nodes is much higher than 5%. Also, at a higher packet expiration time of 30 min, a cheating node attracts

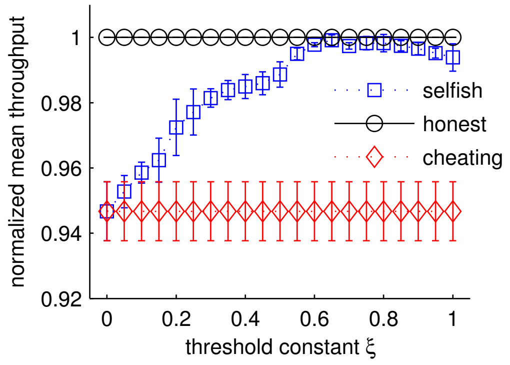

Figure 6: CRISP is robust over different values of threshold constant ξ – at all values of ξ throughput with selfish behavior is greater than that of cheating

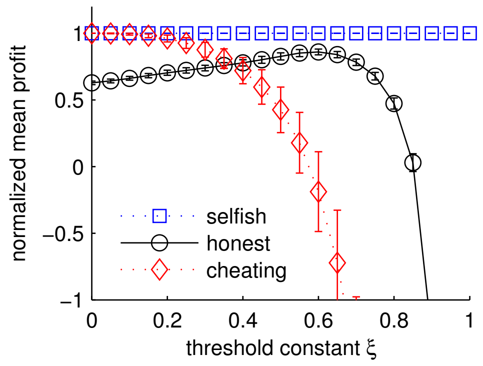

Figure 7: CRISP is incentive compatible over different values of threshold constant ξ – at values $\xi > 0.4$ profit of selfish (or honest) behavior is greater than cheating

more packets than it does at the expiration time of 15 min and less. Therefore, the throughput of the network decreases from 5% to 8% with high packet expiration times.

7. BACKGROUND AND RELATED WORK

In ad hoc networks, the earliest work proposes the use to a virtual currency 'nuglets' that are put with the generated packet and each relay on the forwarding path takes its share [6]. However, this scheme requires tamper–proof hardware. An elegant solution is proposed in [34] that provides credit to relays that forward a packet. The payment for source and relays is adjusted so that following the forwarding protocol is in their best interest. This solution, however, does not result in optimal selection of routing path. Therefore, a VCG mechanism based technique is proposed to select optimal routing paths as being truthful maximizes a nodes reward [1, 35]. [31] also considers the source with limited budget and provides a mechanism that makes routing/forwarding a Nash equilibrium strategy.

In opportunistic networks, [22] requires the relay to pay for receiving a packet, whereas it gets its reward by requesting a higher payment from the next relay. The exact amounts of payments are not discussed and the scheme requires a credit authority to be online in order to resolve payment conflicts. [20, 37] introduce the concept of layered coins for payments such that the credit authority never needs to be online at the time of transaction. The algorithm proposed in [8] secures information of each encounter in an un–forgeable

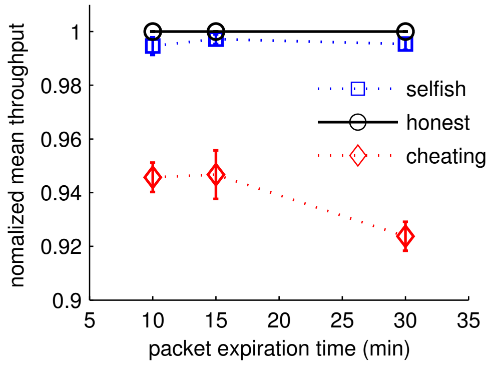

Figure 8: Throughput across different packet expiration times

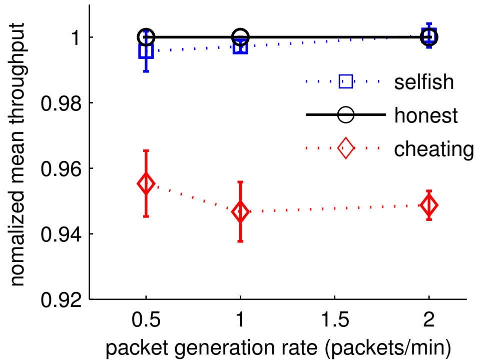

Figure 9: Throughput across different packet generation rates

ticket that is signed by private keys of encountering nodes. These tickets are then used to estimate delivery likelihood of nodes in [17]. While [17] secures the routing metric, a relay can still drop all the packets it receives using its true routing metric, and there is no check on that. [24] secures this limitation in [17], by recording the information about exchanged packets in encounter tickets, whereas, [18] address same limitation by using ACK packets and a more direct reputation–assisted forwarding mechanism.

Works in [1, 31, 35] still assume that nodes do not collude to increase their reward. [36] provides a mechanism to defend against collusion when colluding nodes do not completely trust each other and [32] shows that no mechanism can defend against collusion when colluding nodes can share each other's profit. Owing to this negative result on defense against collusion, [2] proposes a framework that monitors the network for special cases of collusion. However, such a framework cannot be used in opportunistic networks because observing network traffic is not possible due to disconnections and sparse connectivity. In opportunistic networks, all these techniques [8, 17, 18, 20, 22, 24, 37] fail when multiple nodes can collude with each other. In collusion, it can be assumed that nodes will share their private keys and earned rewards with each other. Therefore, some relays can add other relays in the path even when others do not take part in actual forwarding [8, 17, 18, 20, 22, 24, 37]. This way nodes can earn additional credit and use it to launch flooding or black–hole attacks. Similarly, colluding nodes can easily forge encounter tickets [8, 17], packet exchange records [24] or reputation values [18] by using their private keys and signing off fake receipts.

8. ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grants ECCS-0824120, CSR 0834493, and CNS-1117866. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the author and do not necessarily reflect the views of the National Science Foundation.

9. CONCLUSIONS

In this paper, we proposed CRISP, a collusion–resistant credit scheme designed to ensure profit maximization when a node behaves honestly to receive and forward packets. Existing credit schemes reward upon forwarding packets which reduces the network throughput when a node tries to maximize its forwarding rate. In comparison, CRISP optimizes a specific packet arrival and forwarding rate by adjusting the ratio of reward to payment (for forwarding and receiving packets respectively). CRISP secures against collusion of nodes to change routing utility or add fake relays in the path and against both flooding and blackhole attacks. Extensive simulation results demonstrate that CRISP promotes honest (or selfish) behavior which maintains good throughput. CRISP is found to be robust with varying mobility characteristics, packet generation rates, packet expiration times and threshold constant. In future work, we plan to evaluate the opportunistic payment framework and extend CRISP for content sharing and service execution problems in opportunistic networks.

10. REFERENCES

[1] L. Anderegg and S. Eidenbenz. Ad hoc-vcg: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *ACM MobiCom*, 2003.

[2] M. Y. Arslan, K. Pelechrinis, I. Broustis, S. V. Krishnamurthy, P. Krishnamurthy, and P. Mohapatra. Detecting route attraction attacks in wireless networks. In *Proc. IEEE MASS*, 2011.

[3] D. Y. Barrer. Queuing with impatient customers and ordered service. *Operations Research*, 5(5):pp. 650–656, 1957.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University, 2004.

[5] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. Maxprop: Routing for vehicle-based disruption-tolerant networks. In *Proc. IEEE INFOCOM*, 2006.

[6] L. Buttyan and J. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM MoNet*, 8(5):579–592, 2003.

[7] M. C. C. Boldrini and A. Passarella. Contentplace: Social-aware data dissemination in opportunistic networks. In *Proc. MSWiM*, 2008.

[8] S. Capkun, L. Buttyan, and J. Hubaux. Sector: Secure tracking of node encounters in multi-hop wireless networks. In *Proc. ACM SASN*, 2003.

[9] B. B. Chen and M. C. Chan. Mobicent: a credit-based incentive system for disruption tolerant network. In *Proc. IEEE INFOCOM*, 2010.

[10] T. Chen and S. Zhong. Inpac: An enforceable incentive scheme for wireless networks using network coding. In *Proc. IEEE INFOCOM*, 2010.

[11] F. C. Choo, M. C. Chan, and E.-C. Chang. Robustness of dtn against routing attacks. In *Proc. COMSNETS*, 2010.

[12] M. Conti and M. Kumar. Opportunities in opportunistic computing. *Computer*, 43(1):42–50, 2010.

[13] M. Groβ and M. Skutella. Generalized maximum flows over time. In *Matheon Preprint 878*, 2012.

[14] A. Heinemann. *Collaboration in Opportunistic Networks.* VDM Verlag, 2007.

[15] P. Hui, J. Crowcroft, and E. Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proc. ACM MobiHoc*, 2008.

[16] D. Lapsley and S. Low. An optimization approach to abr control. In *IEEE ICC*, 1998.

[17] F. Li, A. Srinivasan, and J. Wu. Thwarting blackhole attacks in disruption-tolerant networks using encounter tickets. In *Proc. IEEE INFOCOM*, 2009.

[18] N. Li and S. K. Das. Radon: reputation-assisted data forwarding in opportunistic networks. In *Proc. MobiOpp*, 2010.

[19] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. In *Proc. SAPIR*, 2004.

[20] R. Lu, X. Lin, H. Zhu, X. Shen, and B. Preiss. Pi: A practical incentive protocol for delay tolerant networks. *IEEE Trans. Wireless Commun.*, 9(4):1483–1493, 2010.

[21] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Games and Economic Behavior*, pages 129–140, 1999.

[22] M. Onen, A. Shikfa, and R. Molva. Optimistic fair exchange in secure forwarding. In *Proc. MobiQuitous*, 2007.

[23] J. Ott, E. Hyytia, P. Lassila, T. Vaegs, and J. Kangasharju. Floating content: Information sharing in urban areas. In *Proc. IEEE PerCom*, 2011.

[24] Y. Ren, M. C. Chuah, J. Yang, and Y. Chen. Detecting blackhole attacks in disruption-tolerant networks through packet exchange recording. In *Proc. IEEE WoWMoM*, 2010.

[25] I. Rhee, M. Shin, S. Hong, K. Lee, and S. Chong. On the levy-walk nature of human mobility. In *Proc. IEEE INFOCOM*, 2008.

[26] I. Rhee, M. Shin, S. Hong, K. Lee, S. Kim, and S. Chong. CRAWDAD trace. `http://crawdad.cs.dartmouth.edu/ncsu/mobilitymodels/GPS/NC_State_Fair`, July 2009.

[27] U. Sadiq and M. Kumar. ProxiMol: Proximity and mobility estimation for efficient forwarding in opportunistic networks. In *Proc. IEEE MASS*, 2011.

[28] M. Shigeno. Maximum network flows with concave gains. *Mathematical Programming*, 107:439–459, 2006.

[29] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Efficient routing in intermittently connected mobile networks: the single-copy case. *IEEE/ACM Trans. Netw.*, 16(1):63–76, 2008.

[30] L. A. Végh. Concave generalized flows with applications to market equilibria. *CoRR*, 2011.

[31] W. Wang, S. Eidenbenz, Y. Wang, and X.-Y. Li. Ours: optimal unicast routing systems in non-cooperative wireless networks. In *Proc. ACM MobiCom*, 2006.

[32] W. Wang and X.-Y. Li. Low-cost routing in selfish and rational wireless ad hoc networks. *IEEE Trans. Mobile Comput.*, 5(5):596–607, 2006.

[33] F. Wu, T. Chen, S. Zhong, L. E. Li, and Y. R. Yang. Incentive-compatible opportunistic routing for wireless networks. In *Proc. ACM MobiCom*, pages 303–314, 2008.

[34] S. Zhong, J. Chen, and Y. Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proc. IEEE INFOCOM*, 2003.

[35] S. Zhong, L. E. Li, Y. G. Liu, and Y. R. Yang. On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks: an integrated approach using game theoretic and cryptographic techniques. *Wirel. Netw.*, 13:799–816, December 2007.

[36] S. Zhong and F. Wu. On designing collusion-resistant routing schemes for non-cooperative wireless ad hoc networks. In *Proc. ACM MobiCom*, 2007.

[37] H. Zhu, X. Lin, R. Lu, Y. Fan, and X. Shen. Smart: A secure multilayer credit-based incentive scheme for delay-tolerant networks. *IEEE Trans. Veh. Technol.*, 58(8), 2009.