Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 569129, 14 pages
http://dx.doi.org/10.1155/2014/569129

Research Article

Node-Dependence-Based Dynamic Incentive Algorithm in Opportunistic Networks

Ruiyun Yu and Pengfei Wang

Software College, Northeastern University, Shenyang 110819, China

Correspondence should be addressed to Ruiyun Yu; yury@mail.neu.edu.cn

Received 8 June 2014; Accepted 7 September 2014; Published 28 September 2014

Academic Editor: Qinggang Meng

Opportunistic networks lack end-to-end paths between source nodes and destination nodes, so the communications are mainly carried out by the "store-carry-forward" strategy. Selfish behaviors of rejecting packet relay requests will severely worsen the network performance. Incentive is an efficient way to reduce selfish behaviors and hence improves the reliability and robustness of the networks. In this paper, we propose the node-dependence-based dynamic gaming incentive (NDI) algorithm, which exploits the dynamic repeated gaming to motivate nodes relaying packets for other nodes. The NDI algorithm presents a mechanism of tolerating selfish behaviors of nodes. Reward and punishment methods are also designed based on the node dependence degree. Simulation results show that the NDI algorithm is effective in increasing the delivery ratio and decreasing average latency when there are a lot of selfish nodes in the opportunistic networks.

1. Introduction

The occurrence of plenty of mobile smart devices equipped with short-range wireless communications boosts the fast rise of opportunistic networks. One of the main features of the opportunistic network is that it does not require a complete path existing from a source node to a destination node. The opportunistic network is a kind of delay tolerant networks (DTNs), and it utilizes communication opportunities obtained from node movement to relay packets.

Opportunistic networks have been widely used in the wildlife tracking [1], pocket switched network [2], automobile network [3], and so forth. The basic routing strategy of opportunistic networks is "store-carry-forward." Data transmission is mainly dependent on intermediate nodes' relay, so node cooperation is a critical factor that deeply affects performance of opportunistic networks. However, researchers have proved that a large amount of nodes would like to gain more and pay less [4], which makes it important to stimulate nodes to cooperate.

There are usually three kinds of nodes in opportunistic networks, including ordinary nodes, malicious nodes, and selfish nodes [5].

A selfish node usually propagates its packets with the help of other nodes, while it refuses to relay other nodes' packets. Such behavior does reduce nodes' own resource consumption, but it decreases the network's delivery ratio and causes longer delivery delay.

A great number of selfish nodes will damage the reliability of networks, and the experiment shows that when the number of selfish nodes increases to 10%–40%, the delivery ratio of the network will decrease by 16%–32% [4].

In order to decrease the number of selfish nodes in a network, a lot of work has been done. The main incentive methods to stimulate selfish nodes are divided into three kinds, which are based on reputation, virtual currency, and gaming, respectively.

1.1. Reputation Based Incentive Algorithms. Marti et al. [4] propose a routing protocol which can detect behaviors of neighbor nodes to improve performance of ad hoc networks. The protocol contains Watchdog and Pathrater. The Watchdog stores node reputation and finds misbehavior nodes, and the Pathrater is used to choose a routing path which does not include misbehavior nodes.

Buchegger and Le Boudec [6, 7] propose the CONFIDANT protocol which evolves from [4] and adds the mechanism that can obtain node's indirect neighborhoods' reputations. The CONFIDANT protocol achieves the goal of isolating and punishing selfish nodes very well.

We also propose a kind of an Accumulated Reputation Model [8] to stimulate the nodes to provide more accurate and truthful data in a Participatory Sensing System.

1.2. Virtual Currency Based Incentive Algorithms. In virtual currency based incentive algorithms each node usually owns equal virtual currency when initializing the network. In order to enhance mutual cooperation, a node will obtain some virtual currency after it helps others to relay packets. Accordingly, when a node asks for assistance from other nodes, it also should pay some virtual currency to them. Buttyan and Hubaux [9] propose a virtual currency based incentive algorithm to promote the cooperation in the network, and this kind of virtual currency is called "Nuglet."

In the domain of electronic commerce involving resource allocation, the well-known Vickrey-Clarke-Groves (VCG) mechanism [10–12] is efficient. A number of incentive mechanisms are proposed by exploiting VCG strategy, such as Ad Hoc-VCG [13], Corsac [14], Team [15], and RPP [16].

1.3. Gaming Based Incentive Algorithms. Gaming based approaches are good ways on modeling incentive processes using classical game theory [17].

The Ad Hoc-VCG algorithm is also a gaming based incentive algorithm. It is a reactive and cost-efficient routing protocol. The nodes are motivated to reveal their true costs for forwarding data. The protocol also guarantees that routing is done along the most cost-efficient path by paying to the intermediate nodes a premium over their actual costs for forwarding data packets.

Srinivasan et al. [18] propose a GTFT model which uses the tit for tat (TFT) strategy to balance profits among nodes. Zhang et al. [19] propose a game based incentive algorithm using four important context parameters and Kalman filtering for prediction.

Dynamic repeated game theory is appropriate to be used in stimulating nodes in opportunistic networks. In dynamic gaming, a node decides to cooperate or not by considering its opponent's last choice. Generally, a node prefers to refuse its opponent when noncooperation has been chosen by the opponent in current gaming phase. However, meeting opportunity is precious in opportunistic networks, so nodes should tolerate a certain level of noncooperation for maintaining good network performance.

In our previous work [20], we designed the original model of node-dependence-based dynamic gaming incentive (NDI) algorithm. We improved the NDI algorithm by using entropy weight method to calculate weights, providing a new reward method and doing more meaningful simulation in this paper.

The NDI algorithm provides a tolerance mechanism in which several times of refusal to relay packets are tolerable, while reward and punishment are also provided based on the node dependence degree.

The remainder of this paper is organized as follows. The system model is illustrated in Section 2; Section 3 specifies the NDI algorithm; extensive simulations have been done for performance evaluation in Section 4; Section 5 concludes the paper.

2. System Model

2.1. Basic Hypothesis

(i) The opportunistic network is modeled as a graph $G = (V, E)$, where $V = (v_1, v_2, \ldots, v_n)$ is the vertex set and E is the edge set. In addition, there exists a communication link if and only if two nodes are within the transmission range.

(ii) The sizes of data packets over the opportunistic network are about the same, and it costs the same in sending and relaying packets.

(iii) The network running time is divided into a series of time slices $t_1, t_2, \ldots, t_n$, and each time slice guarantees that a packet can reach to another relay node or the destination node.

(iv) All nodes can be trusted, with using existing trust models [21, 22]. Other reliability problems are not considered in this paper.

2.2. Node Data Structure. Each node i stores information of every other node j and updates it upon receiving a response to its relay request. The items of the data structure of node i is shown Table 1.

Description of each column of the data structure is detailed in the table. N_{ij} is not the total times that node j rejects node i, whereas it is just a relative value. When it is bigger than T_{ij}, node j will be punished by node i (see Section 3).

3. Algorithm Design

In this section, the node active coefficient (NAC) and the node isolation coefficient (NIC) are defined firstly, and then the node dependence degree is calculated on the basis of NAC, NIC, and T_{ji}. The node-dependence-based dynamic gaming incentive (NDI) algorithm is finally elaborated.

3.1. Node Dependence Degree

3.1.1. Node Active Coefficient. The node active coefficient (NAC) is the probability of a node to meet other nodes. The bigger the NAC is, the more the nodes it probably meets, and hence it has more opportunities to request others for relaying packets. The fuzzy set is exploited to define NAC, and its membership function is shown as formula (1). Consider

$$\alpha = \left[\alpha' + \left(1 - \alpha'\right)\frac{x}{m}\right]\gamma, \tag{1}$$

where α is the NAC value and α' is the previous NAC value before updating. The number of nodes in the network is set to

Table 1: Data structure for node i.

Columns	Description
j	Identification of j
N_{ij}	Relative times that i has been rejected by j
T_{ij}	Tolerance times that i is rejected by j
F_{ij}	Flag for whether i is punishing j or not, Boolean type; flag = 1 means punishing
N_{ji}	Relative times that j has been rejected by i
T_{ji}	Tolerance times that j is rejected by i
F_{ji}	Flag for whether j is punishing i or not, Boolean type; flag = 1 means punishing.

m, x is the number of nodes that the node met during the last time period, and γ is an aging factor which falls in the range $[0, 1]$. Each node updates its NAC after a period of time by formula (1). When a node initializes, α and α' will be set to 0.5 which means the node is in a fuzzy state. The NAC value α will tend to be active or inactive through adjusting those parameters.

3.1.2. Node Isolation Coefficient. The node isolation coefficient (NIC) is the level of isolation in the network. If a node performs selfishly to another node for a number of times, it will be isolated by the node. The NIC value σ is calculated as follows:

$$\sigma = \frac{f+1}{g+1}, \tag{2}$$

where g is the total number that a node meets and f is the number of the nodes which are isolating the node.

If a node accumulated a high NIC value, the nodes which are isolating it will refuse to relay its packets even though it has a high NAC value.

3.1.3. Node Dependence Degree. The node dependence degree (NDD) is the degree to which a node relies on another node.

It is widely acknowledged that the communications are usually fulfilled through multiple hops in opportunistic networks. The source node might never meet the destination node, so the source node also cannot know exactly which intermediate nodes relay the packets to the destination node. It is hard to calculate NDD by the records of successful relay and hit on destination, because it has no idea which node helped it. Here, we approximately calculate NDD through NAC, NIC, and T_{ji}. High NAC and T_{ji} will lead to low NDD, and similarly high NIC results in high NDD.

An entropy weighted method is proposed to calculate NDD with different weights. NDD is computed through current and historical values of NAC, NIC, and T_{ji}. The entropy weighted method is a method determining weights objectively.

Firstly, a 3×3 evaluation matrix is used to record NAC, NIC, and T_{ji} in recent three times. The evaluation matrix is shown as (3). The first row stands for values that the node possesses currently.

Consider

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}. \tag{3}$$

And each row is valued through (4), where w ranges from 1 to 3.

Consider

$$\begin{aligned} r_{w1} &= (1-\alpha)_w, \\ r_{w2} &= \sigma_w, \\ r_{w3} &= \left(\frac{1}{\sqrt{T_{ji}}} \right)_w. \end{aligned} \tag{4}$$

The entropy method is used to obtain the weight of each column in matrix R.

Consider

$$z_{wv} = \frac{r_{wv}}{\sum_{w=1}^{3} r_{wv}}. \tag{5}$$

Firstly, z_{wv} is defined using (5), where v varies from 1 to 3. Then, entropy for each column can be calculated as shown in (6), where $c = 1/\ln 3$.

Consider

$$e_v = -c \sum_{w=1}^{3} z_{wv} \ln z_{wv}. \tag{6}$$

Define $W = (w_1 \;\; w_2 \;\; w_3)$ as the weight vector; each element is calculated by

$$w_v = \frac{(1-e_v)}{\sum_{v=1}^{n} (1-e_v)}. \tag{7}$$

The current NDD value Ψ can be obtained through

$$\text{NDD} = W \circ R_1^T = (w_1 \;\; w_2 \;\; w_3) \circ (r_{11} \;\; r_{12} \;\; r_{13})^T. \tag{8}$$

Because the weight should be calculated through three-time contact of two nodes, the weights are initialized to 1/3 in the first two contacts.

3.2. Dynamic Gaming. The process of requesting to relay packets between two nodes can be regarded as a gaming process.

Each node has a strategy space $S = \{\text{C}, \text{D}\}$, where C represents "accept relay request of its opponent," and D stands for "refuse to relay." During every stage of the gaming process, a node will choose a strategy from its strategy space. For a one-stage gaming, the two counterparts only care about one-time profit, and the payoff matrix is shown in Table 2.

If two nodes do not cooperate with each other, they have no benefit or consumption. In such case, both parts acquire 0 profits.

Apparently, when two nodes only care about one-time profit they will reach to $\{\text{D}, \text{D}\}$, which is not expected to

Table 2: Payoff matrix for one-stage gaming.

Strategy	j: C	j: D
i: C	$i = v - c$, $j = v - c$	$i = -c$, $j = v$
i: D	$i = v$, $j = -c$	$i = 0$, $j = 0$

c is the cost for helping a node to relay a packet, including resources consumptions and so forth; the benefit one node obtains because of opponent's cooperation is represented by v.

see in an opportunistic network. All nodes choosing not to cooperate will ruin the network.

Fortunately a repeated gaming is a good choice to reach a subgame perfect Nash equilibrium and achieve mutual cooperation [21].

In this paper, the node-dependence-based dynamic gaming incentive (NDI) algorithm is proposed which stimulates nodes to cooperate on data forwarding by elaborate design of a subgame perfect Nash equilibrium.

Node j chooses to accept or refuse upon receiving a packet relay request from node i while node i will consider whether to reward or punish node j according to its response based on the node dependence degree (NDD).

In NDI, node i rewards or punishes node j by increasing or decreasing tolerance times T_{ij}. If the punishment tends to high, a selfish node will be likely to suspend its selfish behavior in time. When a node is highly dependent on another one, it will reward the node properly.

If a node is a common node, it will unconditionally help others to relay packets, unless it has objective reasons to refuse, such as node j is transmitting data, buffer is full, and so forth. If a node is a selfish node, when it wants to reject other nodes' request, it should consider its condition and whether other nodes will punish it or not.

Let $U(\infty, \delta)$ stand for an infinite repeated gaming process between node i and node j, where δ is a weight coefficient to indicate the extent of addressing future profit. The bigger the δ is, the more the node pays attention to future profit. Table 3 shows the profit sequence for node j when both counterparts choose to cooperate [23].

The profit that a common node j obtains is calculated as formula (9). Consider

$$(v - c) + \delta (v - c) + \delta^2 (v - c) + \cdots = \frac{v - c}{1 - \delta}. \tag{9}$$

Similarly, the profit that a common node i obtains is also $(v - c)/(1 - \delta)$. If the profit of a node is over $(v - c)/(1 - \delta)$, it means that the node has chosen to be selfish during previous gaming. If node i tolerated node j's noncooperation for T_{ij} times in an infinite repeated gaming, the maximum total profit of node j would reach to $((v - c)/(1 - \delta)) + c(\underbrace{\delta^{q1} + \delta^{q2} + \cdots + \delta^{qx}}_{T_{ij}})$ ($q_1, q_2, \ldots, q_x$ are the superscripts of profit items when node j rejects node i). Based on the profit of node j, we can adjust the maximum total profit of node j by changing T_{ij}. The NDI algorithm is detailed in Algorithm 1.

Where $f(T_{ij})$ is the punishment function and $g(T_{ij})$ is the reward function. The intensity of punishment method is $T_{ij} - f(T_{ij})$, and the intensity of reward method is $g(T_{ij}) - T_{ij}$.

The punishment function should rely on NDD. If NDD is big, it means "I rely on you a lot"; the punishment strength should be lower. On the contrary, if NDD is small, it means "I rely on you little"; the punishment strength can be higher than with bigger NDD. The reward function actually should be steadier than punishment method. In practice, the cooperation should be continuous and steady. A node also should consider the NDD value. The bigger the NDD is, the more heavily it should be rewarded by the other one.

Based on the above analysis, the punishment function and the reward function should meet the following conditions.

(1) The punishment is designed to be heavier than the rewarding which helps to keep the balance of cooperation, so the condition $(T_{ij} - f(T_{ij})) > (g(T_{ij}) - T_{ij})$ should be met.

(2) In order to stimulate nodes to help other relay packets, approaches for rewarding nodes are proposed. T_{ij} should increase steadily, when counterparts cooperate with each other for a long time. In other words, $g(T_{ij})$ should be a linear function to ensure the increase of T_{ij}.

(3) Accordingly, to punish nodes which would not like to cooperate with each other, the function $f(T_{ij})$ should lower others' T_{ij}. The punishment method follows the principle that the more I "trust" you, the more the intensity of punishment is if you betray me. The intensity of punishment will be heavier with the increase of T_{ij}.

Based on the above analysis, the punishment method is designed as formula (10), and the reward function is designed as formula (11).

Because NDD varies from 0 to 1, the level of reward and punishment can be adjusted according to the NDD value. Consider

$$f\left(T_{ij}\right) = \left\lfloor (1 + \Psi) \sqrt{T_{ij}} \right\rfloor, \tag{10}$$

$$g\left(T_{ij}\right) = \left\lfloor 2\Psi + T_{ij} \right\rfloor. \tag{11}$$

Figure 1(a) depicts formula (10) with different Ψ. It is clear to conclude that T_{ij} will be decreased after being recalculated by $f(T_{ij})$, which satisfies the abovementioned conditions.

The initial value of T_{ij} should be set by considering different conditions of opportunistic networks. And T_{ij} is usually set as 4, and the value of $1/\sqrt{T_{ji}}$ equals 0.5.

The punishment intensity with different Ψ is shown in Figure 1(b). With the increase of T_{ij}, the intensity of punishment is also improved faster. It meets the principle that the more I "trust" you, t the more the intensity of punishment will be if you betray me.

From Figure 1(b), it is verified that the smaller the node dependence degree (NDD) is, the heavier the punishment intensity is. The punishment intensity is higher with the increasing of T_{ij} and Ψ.

Formula (11) with different Ψ is depicted in Figure 1(c), which also satisfies the conditions proposed above.

Table 3: Profit sequence for node j.

Stage	1	2	⋯	T	...
Choice set	{C, C}	{C, C}	⋯	{C, C}	{C, C}
Profit for node j	$v - c$	$v - c$	⋯	$v - c$	$v - c$

(1) Initialize the parameters in Table 1 and the weight coefficient δ when nodes encounter at the first time, and Calculate NAC, NIC, and NDD.
(2) Node i asks node j for relaying a packet, and waits for j's response.
(3) **if** j rejects i
(4) **if** F_{ij} == false
(5) $N_{ij} \leftarrow N_{ij} + 1$
(6) **end if**
(7) **if** F_{ij} == false and $N_{ij} \geq T_{ij}$
(8) $F_{ij} \leftarrow$ true
(9) $T_{ij} \leftarrow f(T_{ij})$
(10) **end if**
(11) **else**
(12) **if** F_{ij} == true
(13) $N_{ij} \leftarrow N_{ij} - 1$
(14) **if** $N_{ij} \leqslant T_{ij}$
(15) $F_{ij} \leftarrow$ false
(16) **end if**
(17) **end if**
(18) **if** *j is not punished for a time period T*
(19) $T_{ij} \leftarrow g(T_{ij})$
(20) **end if**
(21) **end if**

Algorithm 1: The Node-dependence-based dynamic gaming incentive algorithm.

$g(T_{ij})$ is a linear function, the reward of each time is 2Ψ, and $g(T_{ij})$ satisfies the fact that T_{ij} should be increase steadily.

In addition, the reward and punishment history may also be leveraged when calculating the $f(\Psi)$ and $g(\Psi)$. This is left to future study.

When a selfish node j is "smart" enough, it will not reject i after rejecting it for T_{ij} times; otherwise, it will be punished by node i and has to return back much profit if it wants to be forgiven by node i. So a "smart" node will choose to cooperate after rejecting another one for T_{ij} times. So a "smart" selfish node will always perform rationally to obtain high profit.

The profit sequence of a "smart" selfish node is shown in Table 4. It shows that T_{ij} decides the maximum m profit that node j could obtain and the maximum loss that node i would suffer.

The maximum loss for node i and the maximum profit for node j can be calculated by formulas (12) and (13), respectively. Consider

$$c + \delta c + \cdots + \delta^{T_{ij-1}} c = \frac{c\left(1 - \delta^{T_{ij}}\right)}{1 - \delta}, \tag{12}$$

$$v + \delta v + \cdots + \delta^{T_{ij-1}} v = \frac{v\left(1 - \delta^{T_{ij}}\right)}{1 - \delta}. \tag{13}$$

When the profit of node j obtained because of selfish behaviors exceeds $v(1 - \delta^{T_{ij}})/(1 - \delta)$, node i will severely punish node j. Whereas a selfish node j has to cooperate when N_{ij} approaches T_{ij} in case it will be punished.

If two nodes encountered for several times and if the condition of Line 6 in Algorithm 1 holds in these rounds, T_{ij} will finally be set to 1. This is a kind of classical gaming method, known as "tit for tat (TFT)" [16]. This method advocates "Deal with a man as he," and it is an alternative to promote long-term cooperation. We simulate NDI and this method for performance comparison in Section 4.3.

Except for punishment, the NDI algorithm also provides a reward mechanism. When node j always chooses to cooperate upon receiving packet relay requests of i for a time period T, node i will reward node j as shown in formula (11). This will encourage nodes to relay packets and hence promote efficiency of the network.

The following will analyze the profits when a node is "smart," "stupid," or "ordinary."

Ignoring the node's attention to the further profits δ, a node's profit mainly relies on the T_{ij}. As said above, maintaining a high T_{ij} is a good strategy to obtain a high profit. However, NDI is an incentive algorithm to stimulate nodes by changing T_{ij}. N_{ij} can stand for the profit that a node obtains.

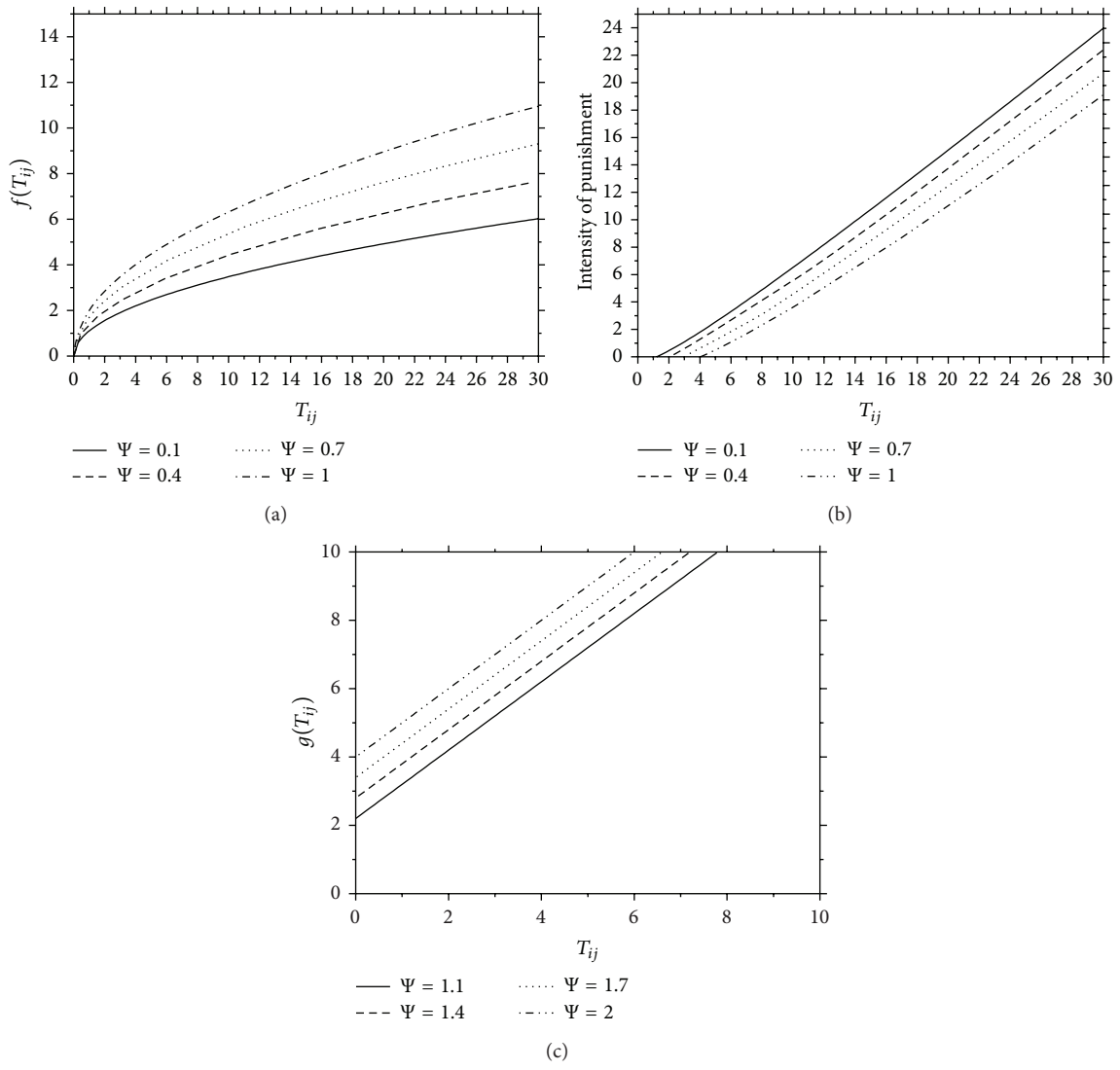

Figure 1: (a) $f(T_{ij})$ with different Ψ. (b) Punishment intensity with different Ψ. (c) $g(T_{ij})$ with different Ψ.

Table 4: Profit sequence when node j is a "smart" selfish node.

Stage	1	⋯	T	⋯	$T + T_{ij}$
Choice	{C, C}	{C, C}	{C, D}	{C, D}	{C, D}
Profit sequence for node i	$v - c$	$v - c$	$-c$	$-c$	$-c$
Profit sequence for node j	$v - c$	$v - c$	v	v	v

A "smart" selfish node would like to reject other nodes' request $T_{ij} - 1$ times, where it can obtain the highest profit and will not be punished at the same time. With the time passing, the profits can be described as in Figure 2(a). The initial T_{ij} is set as 8 in Figure 2(a), and the NDI algorithm is added in it.

A "smart" selfish node will not be punished because it always rejects others $T_{ij} - 1$ times. However, the T_{ij} will be increased because of the rewarding of long time cooperation.

A "stupid" selfish node will not cooperate with others at any time, and the profit of a "stupid" selfish node is shown in Figure 2(b).

As shown in Figure 2(b), this kind of selfish node hardly obtains profit from others. So it must choose cooperation to improve its T_{ij} which can improve its profit.

An "ordinary" selfish node is a kind of node that acts more in reality; it will firstly choose noncooperation. After finding the high decreasing of profit, it will try its best to improve its

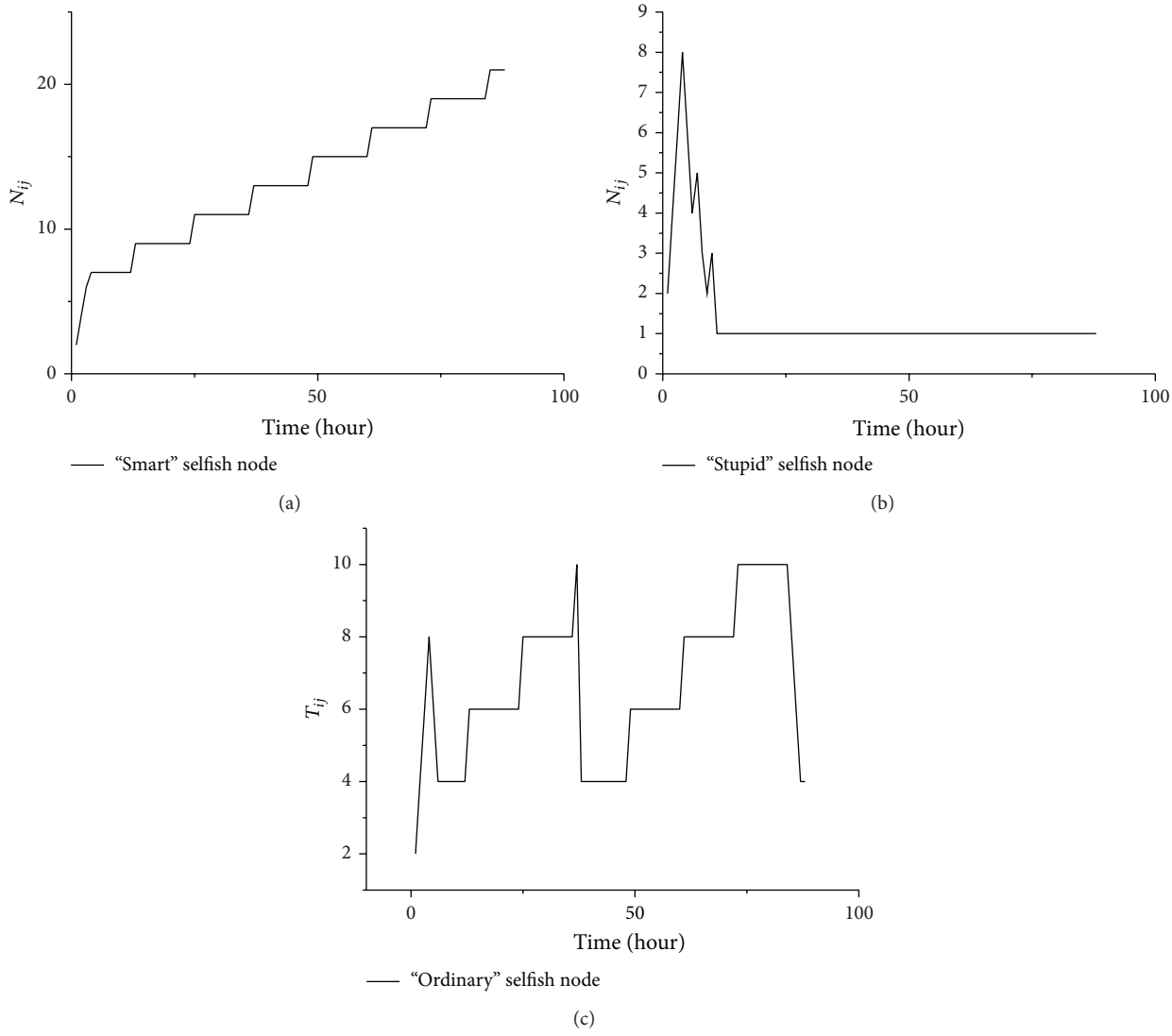

Figure 2: (a) The profit of "smart" selfish node with time increasing. (b) The profit of "stupid" selfish node with time increasing. (c) The profit of "ordinary" selfish node with time increasing.

profit and maintain it. The profit of an ordinary selfish node can be described as in Figure 2(c).

The profit of an "ordinary" node firstly increases quickly, and then the node will be punished. After that, the "ordinary" node notices that it must cooperate to improve its profit, so it will choose to help others to relay packets.

In conclusion, if a node chooses to cooperate, it will obtain a higher profit than these which chose not to cooperate.

4. Simulation Analysis

Simulations are run on the opportunistic networking environment (ONE) [24, 25] which is developed by Helsingin Yliopisto. ONE simulator is a simulator designed especially for opportunistic networks.

4.1. Evaluation Criteria. In order to verify the effectiveness of the NDI algorithm, delivery ratio (*delivery_ratio*), delivery delay (*delivery_delay*), and average hop count (*hopcount_avg*) are chosen for evaluation criteria.

The delivery ratio is the ratio between the number of packets which are delivered to the destination nodes successfully and the number of all packets that are created in networks. This is a critical criterion to value the performance of a routing algorithm [26]. The delivery ratio (*delivery_ratio*) is calculated by formula

$$delivered\_prob = \frac{delivered}{created}, \tag{14}$$

where *delivered* is the number of packets that have been delivered successfully and *created* is the number of packets that has been created in the network.

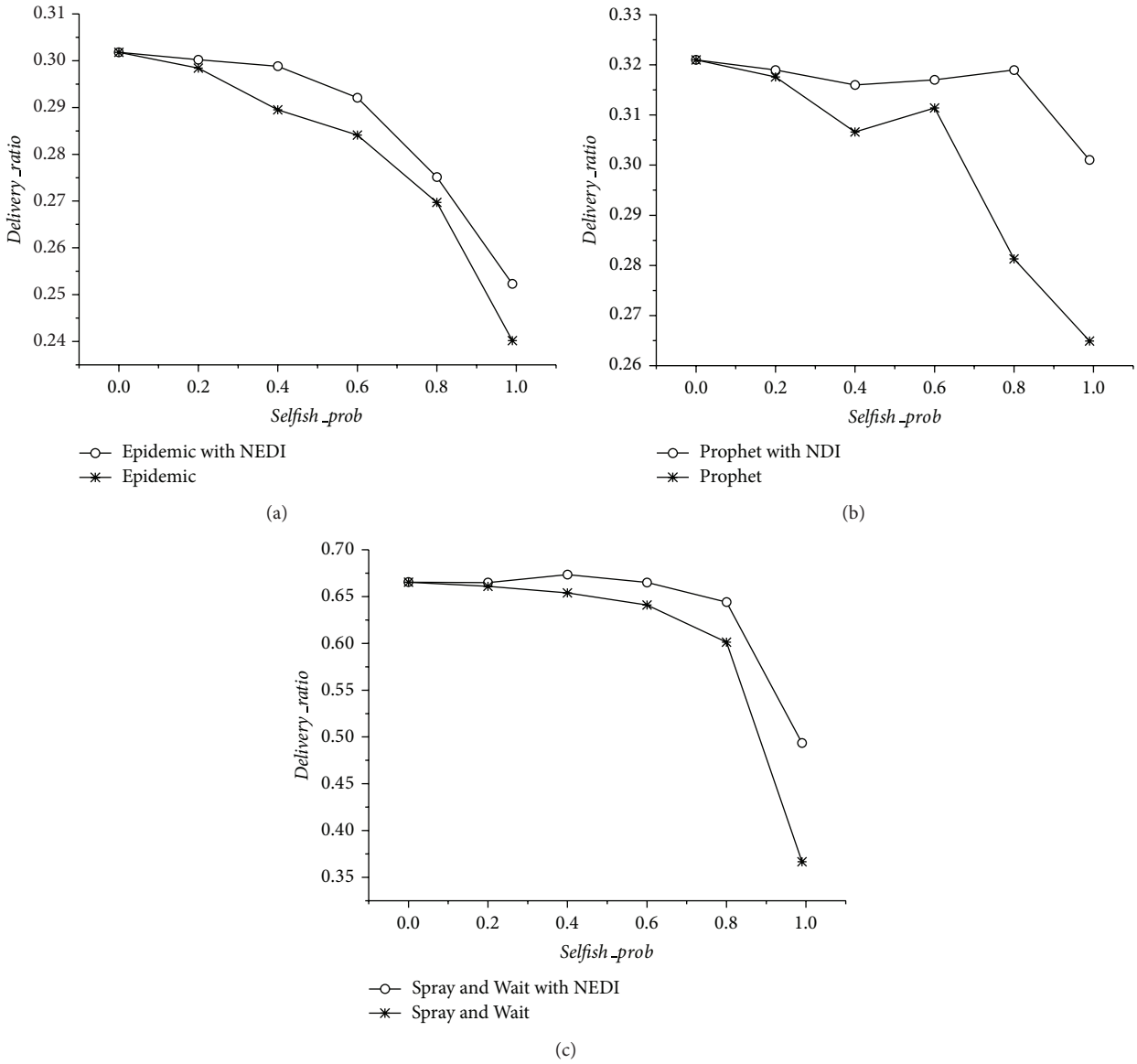

Figure 3: (a) The delivery ratio of Epidemic with and without NDI. (b) The delivery ratio of Prophet with and without NDI. (c) The delivery ratio of Spray and Wait with and without NDI.

Delivery delay is the time period from when a packet is created to when it is delivered successfully. Because environments of networks are different and delivery delays of all packets are not the same, the average delivery latency (*latency_avg*) is used to evaluate the algorithm performance. The smaller the average delivery latency is, the better the performance the algorithms have.

Hop count is the number of relay nodes on the path from the source to the destination. Average hop count (*hopcount_avg*) is used to value performance of networks, and the smaller the better.

To simulate the impact of nodes' selfish behaviors on evaluation criteria (delivery ratio, delivery delay, and average hop count), the concept of selfish probability (*selfish_prob*) is put forward. The selfish probability is the probability of nodes choosing to reject packet relay requests.

4.2. Simulation I

4.2.1. Simulation Setup. Three routing algorithms including Epidemic [27], Prophet [28], and Spray and Wait [29] are used in simulation I to simulate the NDI algorithm.

The Helsinki city is chosen as the scene, where there are 160 mobile nodes divided into two groups. Each node is equipped with a Bluetooth device, and the communication radius of nodes is set to 10 meters. The time period T for rewarding a node is 6 hours. More settings can be found in Table 5.

4.2.2. Delivery Ratio. Simulations are carried out at different selfish probabilities using Epidemic, Prophet, and Spray and Wait with or without NDI algorithm, respectively.

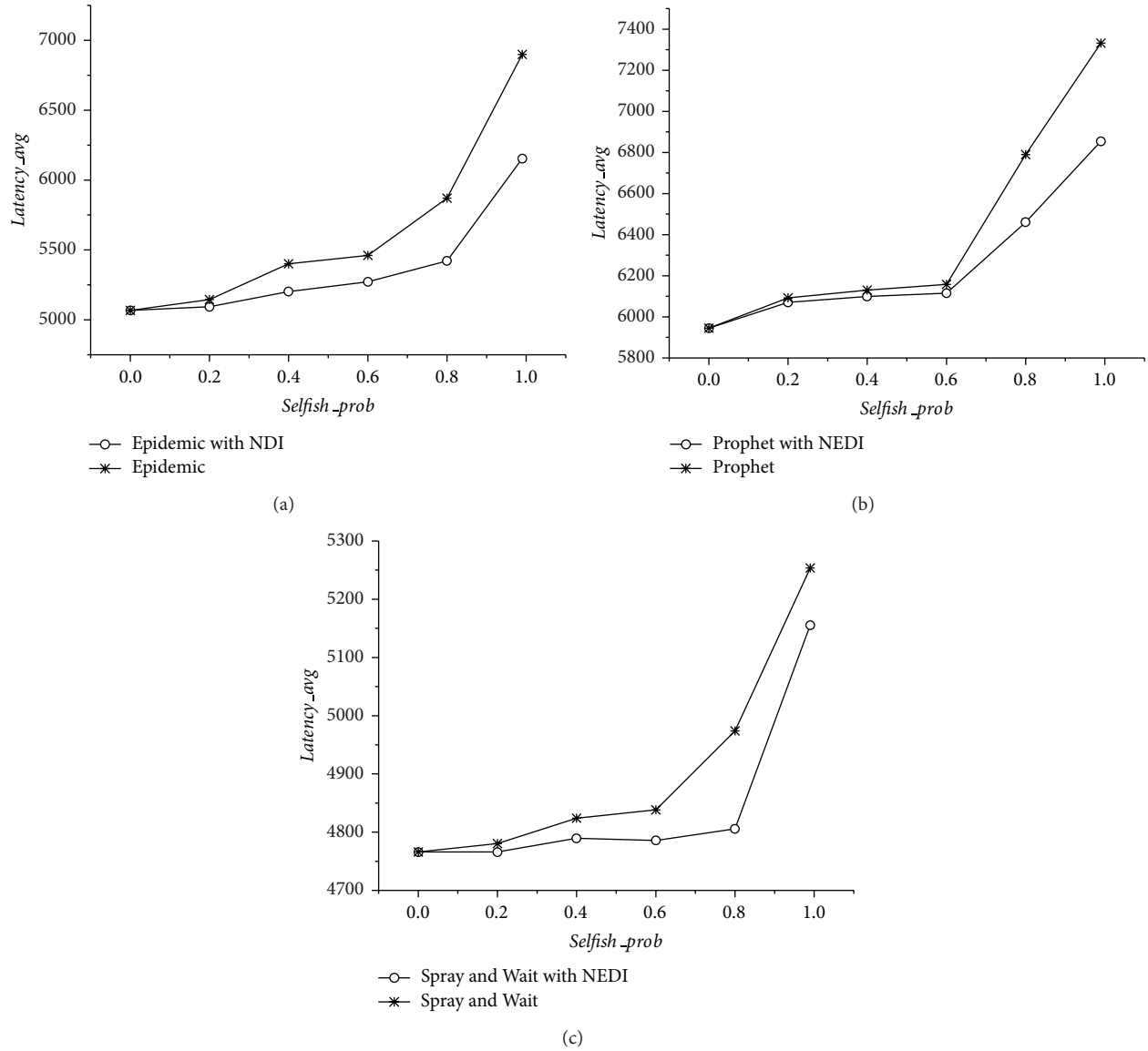

Figure 4: (a) The average latency of Epidemic with and without NDI. (b) The average latency of Prophet with and without NDI. (c) The average latency of Spray and Wait with and without NDI.

Table 5: Parameter settings for simulation I.

Type	Parameter	Value
Scene features	Simulation time	24 h
	Field area	4500 m ∗ 3400 m
	Scene	Helsinki
Node features	Mobility model	SPMBM
	Movement speed	0.5–1.5 m/s
	Transmission rate	250 KB/s
	Maximum transmission range	10 m
	Transmission mode	Broadcast
	Cache size	10 MB
Message features	Packet size	500 KB–1 MB at random
	TTL	5 h
	Frequency of creating packets	From 25 s to 35 s at random

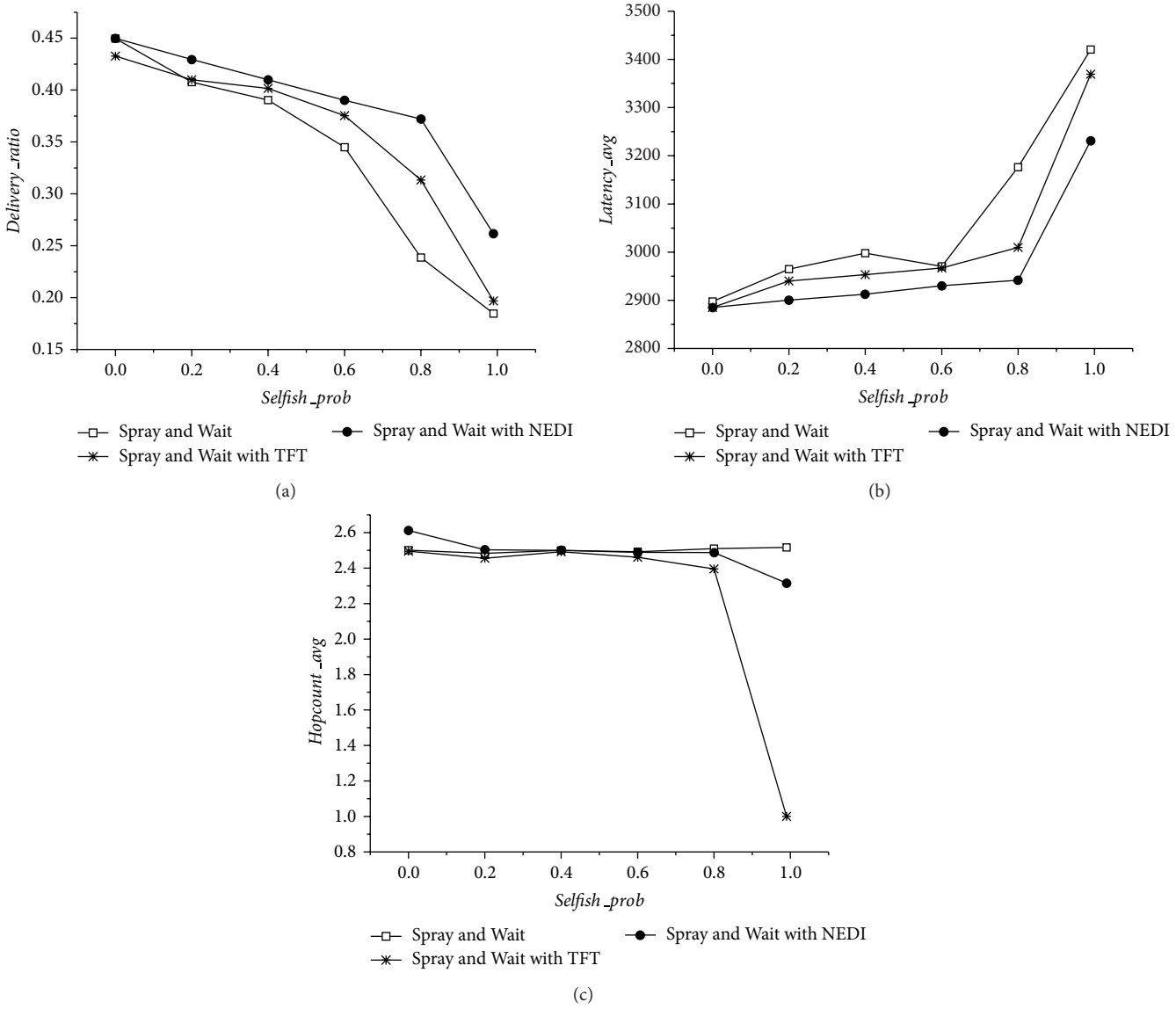

Figure 5: (a) The delivery ratio of three algorithms. (b) The average latency of three algorithms. (c) The average hop count of three algorithms.

The delivery ratio at different selfish probabilities is shown in Figure 3.

The simulations show that the NDI algorithm increases the packet delivery ratio in all three routing algorithms at almost every selfish probability, especially in the Spray and Wait scheme. While the selfish probability increases, the improvement tends to be higher.

The Spray and Wait routing algorithm limits the packet copies, so the NDI algorithm does not have so much effect on it; however, it still can get remarkable improvement on the delivery ratio when the selfish probability is over 60%.

4.2.3. Delivery Delay. The delivery delay of three routing algorithms with and without the NDI algorithm at different selfish probabilities is shown in Figure 4.

As mentioned above, the average latency averages the delay of all successful routing. When the selfish probability increases, the average latency increases accordingly. In all three routing schemes, the NDI algorithm makes them more efficient on delivering packets to the destinations. It is obvious that the NDI algorithm presents better performance along with the selfish probability increasing.

The curves of routing schemes with NDI are much flatter than the original ones, which show that the NDI algorithm has positive effect on lowering transmission latency of packet no matter how selfish the nodes are.

4.3. Simulation II. From simulation I, the Spray and Wait routing algorithm exhibits better performance than the other

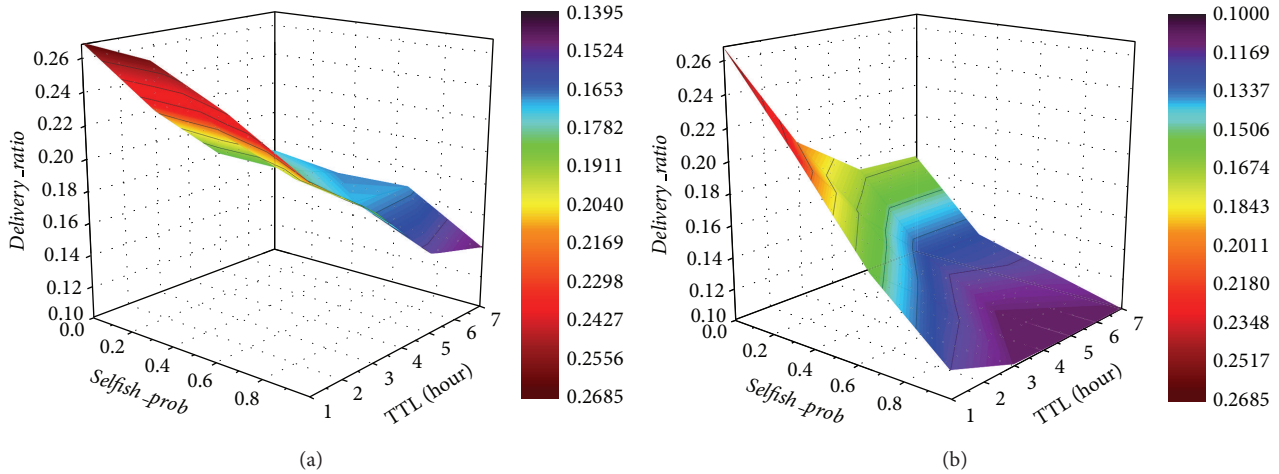

FIGURE 6: (a) The relationship among selfish probability, TTL, and delivery ratio with NDI. (b) The relationship among selfish probability, TTL, and delivery ratio without NDI.

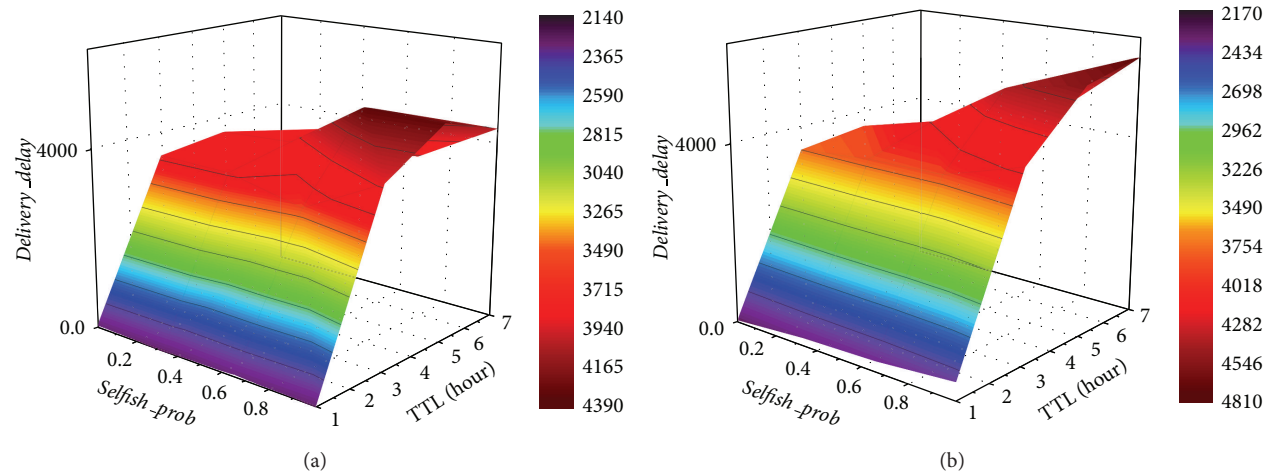

FIGURE 7: (a) The relationship among selfish probability, TTL, and delivery delay with NDI. (b) The relationship among selfish probability, TTL, and delivery delay without NDI.

two ones, so we use simulate the Spray and Wait routing algorithm in simulation II. The TFT method [16] is introduced for performance comparison.

4.3.1. Simulation Setup. The Helsinki city is also chosen as the scene, where there are 160 mobile nodes divided into four groups, including 3 pedestrian groups and a car group. Each node is equipped with a Bluetooth device, and the time period T for rewarding a node is 6 hours. More settings are detailed in Table 6.

4.3.2. Delivery Ratio. Simulations are run under different selfish probabilities which are 0, 0.2, 0.4, 0.6, 0.8, and 0.99, respectively. The delivery ratios of the Spray and Wait algorithm and the other two variations (with NDI and TFT) are shown in Figure 5(a).

The Spray and Wait with NDI algorithm is much better than the original one and the TFT variation on packet delivery probability. The Spray and Wait with TFT algorithm lowers down the delivery ratio when the selfish probability is zero. TFT algorithm will result in a worse performance than the NDI algorithm when selfish probability gets higher.

The reason lies in the "Deal with a man as he" mechanism of TFT algorithm. If a node is rejected by its opponent, it will immediately punish the opponent by rejecting any relay request. The NDI algorithm overcomes this drawback and hence performs much better than the TFT scheme.

4.3.3. Delivery Delay. The average latency at different selfish probabilities for each algorithm is shown in Figure 5(b). The Spray and Wait with NDI algorithm has an optimization compared with TFT algorithm on average latency, while the Spray and Wait with TFT algorithm incurs a little higher latency than that with NDI algorithm when the selfish probability increases. This is also an effect of the "tit for tat." Higher

Table 6: Parameter settings for simulation II.

Type	Parameter	Value
Scene features	Simulation time	12 h
	Field area	4500 m ∗ 3400 m
	Scene	Helsinki
Node features	Mobility model	SPMBM
	Movement speed for cars	2.7–13.9 m/s
	Movement speed for pedestrian	0.5–1.5 m/s
	Transmission rate	250 KB/s
	Maximum transmission range	10 m
	Transmission mode	Broadcast
	Cache size	5 MB
Message features	Packet size	500 KB–1 MB at random
	TTL	5 h
	Frequency of creating packets	From 25 s to 35 s at random

Table 7: Parameter settings for simulation III.

Type	Parameter	Value
Scene features	Simulation time	12 h
	Field area	4500 m ∗ 3400 m
	Scene	Helsinki
Node features	Mobility model	SPMBM
	Movement speed for cars	2.7–13.9 m/s
	Movement speed for pedestrian	0.5–1.5 m/s
	Transmission rate	250 KB/s
	Maximum transmission range	10 m
	Transmission mode	Broadcast
	Cache size	100 MB
Message features	Packet size	500 KB–1 MB at random
	Frequency of creating packets	From 25 s to 35 s at random

chance of nodes' rejection on packet relay inevitably leads to higher latency when delivering packet to its destination.

4.3.4. Average Hop Count. The average hop count for each algorithm is shown in Figure 5(c). When the selfish probabilities increase, the average hop count of packed delivery is slightly smaller in the NDI and TFT variations of the Spray and Wait algorithm.

The most attractive point is when selfish probability is 0.99; the average hop count of the Spray and Wait with TFT algorithm is approximated to 1. The reason is that when the selfish probability is big enough, nodes would not like to relay packets for other nodes and most nodes will deliver their packets directly to the destinations. However, the Spray and Wait with NDI algorithm has adequate tolerance to the rejection of relay requests, so it performs well in this scenario.

4.4. Simulation III. The Epidemic routing algorithm [23] is simulated for performance measure in simulation III, because it does not limit the number of message copies.

4.4.1. Simulation Setup. There are 160 mobile nodes which are divided into four groups, including 3 pedestrian groups and a car group. Each node is equipped with a Bluetooth chip, and the time period T for rewarding a node is 3 hours. More settings are detailed in Table 7.

4.4.2. Delivery Ratio. The relationship among selfish probability, TTL, and delivery ratio is investigated, and the result is shown in Figure 6. Due to the fixed size of cache, the delivery ratio decreases even though the TTL is increasing. However, the approach with NDI performs better than the one without NDI on the descending rate. Simulation results also show that higher selfish probability leads to less delivery ratio.

4.4.3. Delivery Delay. Figure 7 shows the relationship among selfish probability, TTL, and delivery delay. As depicted in Figures 7(a) and 7(b), TTL affects the delivery delay very much. Larger TTL causes higher delay. When the selfish probability increases, the approach with NDI can suppress the delay increase compared to the one without NDI.

5. Conclusion

To stimulate nodes to cooperate on data forwarding in opportunistic networks, the node-dependence-based dynamic gaming incentive (NDI) algorithm is proposed by elaborate design of a subgame perfect Nash equilibrium in a dynamic repeated gaming process. The NDI algorithm provides a mechanism where several times of refusal to relay packets are tolerant, and reward and punishment methods are also designed based on the node dependence degree.

Extensive simulations are implemented for performance evaluation. The results show that the NDI algorithm is effective in increasing the delivery ratio and decreasing average latency when there are a lot of selfish nodes in the opportunistic networks.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant no. 61272529 and the Fundamental Research Funds for the Central Universities under Grants no. N120417002 and no. N130817003.

References

[1] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.-S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 96–107, New York, NY, USA, October 2002.

[2] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference environments," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN '05)*, pp. 244–251, Philadelphia, Pa, USA, August 2005.

[3] B. Hull, V. Bychkovsky, Y. Zhang et al., "CarTel: a distributed mobilesensor computing system," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, pp. 125–138, 2006.

[4] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM '00)*, pp. 255–265, Boston, Mass, USA, August 2000.

[5] C.-L. Chiou and L.-J. Chen, "An evaluation study of routing reliability in opportunistic networks," in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '08)*, pp. 455–456, Hong Kong, May 2008.

[6] S. Buchegger and J. Y. Le Boudec, "Performance analysis of the CONFIDANT protocol (Co-operation of nodes: fairness in dynamic Ad Hoc networks," in *Proceedings of the IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing*, pp. 226–236, Lausanne, Switzerland, 2002.

[7] S. Buchegger and J.-Y. Le Boudec, "Self-policing mobile ad hoc networks by reputation systems," *IEEE Communications Magazine*, vol. 43, no. 7, pp. 101–107, 2005.

[8] R. Yu, R. Liu, X. Wang, and J. Cao, "Improving data quality with an accumulated reputation model in participatory sensing systems," *Sensors*, vol. 14, no. 3, pp. 5573–5594, 2014.

[9] L. Buttyan and J. P. Hubaux, "Nuglets: a virtual currency to stimulate cooperation in self-organized mobile Ad Hoc networks," Tech. Rep. DSC/2001/001, Swiss Federal Institute of Technology (EPFL), 2001.

[10] W. Vickrey, "Counterspeculation, auctions, and competitive sealed tenders," *Journal of Finance*, vol. 16, no. 1, pp. 8–39, 1961.

[11] E. H. Clarke, "Multipart pricing of public goods," *Public Choice*, vol. 11, no. 1, pp. 17–33, 1971.

[12] T. Groves, "Incentives in teams," *Econometrica*, vol. 41, pp. 617–631, 1973.

[13] 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 the 9th Annual International Conference on Mobile Computing and Networking (MobiCom '03)*, pp. 245–259, ACM Press, New York, NY, USA, September 2003.

[14] 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," *Wireless Networks*, vol. 13, no. 6, pp. 799–816, 2007.

[15] J. Cai and U. Pooch, "Play alone or together—truthful and efficient routing in wireless ad hoc networks with selfish nodes," in *Proceedings of the IEEE International Conference on Mobile Ad-Hoc and Sensor Systems (MASS '04)*, pp. 457–465, Washington, DC, USA, October 2004.

[16] M.-Y. Wu and W. Shu, "RPF: a distributed routing mechanism for strategic wireless ad hoc networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '04)*, pp. 2885–2889, Washington, DC, USA, December 2004.

[17] D. Fudenberg and J. Tirole, *Game Theory*, MIT Press, Boston, Mass, USA, 1991.

[18] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao, "Cooperation in wireless ad hoc networks," in *Proceedings of the 22nd Annual Joint Conference on the IEEE Computer and Communications Societies*, pp. 808–817, April 2003.

[19] C. Zhang, Q.-S. Zhu, and Z.-Y. Chen, "Game-based data-forward decision mechanism for opportunistic networks," *Journal of Computers*, vol. 5, no. 2, pp. 298–305, 2010.

[20] R. Yu, P. Wang, and Z. Zhao, "NDI: node dependence-based dynamic gaming incentive algorithm in opportunistic networks," in *Proceedings of the 23rd International Conference on Computer Communications and Networks (ICCCN '14)*, pp. 581–588, Shanghai, China, 2014.

[21] J. Jiang, G. Han, F. Wang, L. Shu, and M. Guizani, "An efficient distributed trust model for wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, 2014.

[22] G. Han, J. Jiang, L. Shu, J. Niu, and H.-C. Chao, "Management and applications of trust in Wireless Sensor Networks: a survey," *Journal of Computer and System Sciences*, vol. 80, no. 3, pp. 602–617, 2014.

[23] C. M. Gui, Q. Jian, H. M. Wang, and Q. Y. Wu, "Repeated game theory based penalty-incentive mechanism in internet-based virtual computing environment," *Journal of Software*, vol. 21, no. 12, pp. 3042–3055, 2010.

[24] TKK/COMNET, 2014, http://www.netlab.tkk.fi/tutkimus/dtn/theone/.

[25] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE simulator for DTN protocol evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, p. 55, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[26] S. Ganeriwa, R. Kumar, and M. B. Srivastava, "Timing-synch protocol for sensor networks," in *Proceedings of the 1st Conference of Embedded Network Sensor Systems*, pp. 138–149, New York, NY, USA, 2003.

[27] V. D. Becker, "Epidemic routing for partially connected ad hoc networks," Proceedings of Technique Report, Department of Computer Science, Duke University, Durham, UK, 2000.

[28] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 3, pp. 19–20, 2003.

[29] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking*, pp. 252–259, ACM, Philadelphia, Pa, USA, August 2005.