IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-2, Issue-9 E-ISSN: 2347-2693

Intelligent Anti-Theft Finding Scheme Towards iTrust Establishment in Delay Tolerant Networks Using VANET

Bhogapathi Swetha[1*], D. Bulla Rao[2] and P. Nageswara Rao[3]

[1*,2,3]*Department Of CSE, Swetha Institute of Technology and Science: Tirupathi*

www.ijcaonline.org

Received: Aug/17/2014 Revised: Aug/26/2014 Accepted: Sep/15/2014 Published: Sep/30/2014

***Abstract*—** Malicious and selfish behaviors represent a serious threat against routing in Delay/Disruption Tolerant Networks (DTNs). Due to the unique network characteristics, designing a misbehavior detection scheme in DTN is regarded as a great challenge. In this paper, we propose iTrust, a probabilistic misbehavior detection scheme, for secure DTN routing towards efficient trust establishment. The basic idea of iTrust is introducing a periodically available Trusted Authority (TA) to judge the node's behavior based on the collected routing evidences and probabilistically checking. A new VANET-based smart parking scheme (SPARK) for large parking lots provide three convenient services for drivers: 1) real-time parking navigation; 2) intelligent anti-theft protection; and 3) friendly parking information dissemination. TA could ensure the security of DTN routing at a reduced cost. To further improve the efficiency of the proposed scheme, we correlate detection probability with a node's reputation, which allows a dynamic detection probability determined by the trust of the users. The extensive analysis and simulation results show that the proposed scheme substantiates the effectiveness and efficiency of the proposed scheme.

***Keywords*—** VANET ,ITRUSTED , DTNS.

I. INTRODUCTION

Delay tolerant networks (DTNs), such as sensor networks with scheduled intermittent connectivity, vehicular DTNs that disseminate location-dependent information (e.g., local ads, traffic reports, parking information) [1], which makes routing quite different from other wireless networks. For example, since an end-to-end connection is hard to setup, store-carry-and forward is used to deliver the packets to the destination. Although many routing algorithms [2]–[3] have been proposed to increase data delivery reliability.

Pocket-switched networks that allow humans to communicate without network infrastructure, are highly partitioned networks that may suffer from frequent dis-connectivity. In DTNs, the in-transit messages, also named bundles, can be sent over an existing link and buffered at the next hop until the next link in the path appears. In DTNs, a node could misbehave by dropping packets intentionally even when it has the capability to forward the data. As a result, in civilian DTNs such as PeopleNet and Pocket Switched Network [4], a node may not be willing to forward packets for others.

To capture user selfishness in a more realistic manner, we have two observations from the social perspective. First, a selfish user is usually willing to help others with whom he has social ties (e.g., friends, coworkers, roommates). Routing misbehavior can be caused by selfish (or rational) nodes that try to maximize their own benefits by enjoying the services provided by DTN while refusing to forward the bundles for others, or malicious nodes that drop packets or modifying the packets to launch attacks.

Corresponding Author: *Bhogapathi Swetha*

Social selfishness will affect node behaviors. As a forwarding service provider, a node will not forward packets received from those with whom it has no social ties, and it gives preference to packets received from nodes with stronger ties when the resource is limited. Thus, a DTN routing algorithm. should take the social selfishness into consideration.

II. PRELIMINARY

2.1 System Model

In this paper, we adopt the system model similar to [5]. We consider a normal DTN consisted of mobile devices owned by individual users. Each node i is assumed to have a unique ID Ni and a corresponding public/private key pair. We assume that each node must pay a deposit C before it joins the network, and the deposit will be paid back after the node leaves if there is no misbehavior activity of the node. Similar to [6], we assume that a periodically available TA exists so that it could take the responsibility of misbehavior detection in DTN. For a specific detection target Ni , TA will request Ni's forwarding history in the global network.

Therefore, each node will submit its collected Ni's forwarding history to TA via two possible approaches. In a pure peer-to-peer DTN, the forwarding history could be sent to some special network components (e.g., roadside unit (RSU) in vehicular DTNs or judge nodes in [7]) via DTN transmission. In some hybrid DTN network environment, the transmission between TA and each node could be also performed in a direct transmission manner (e.g., WIMAX or cellular networks [8]). We argue that since the misbehavior detection is performed periodically, the message

transmission could be performed in a batch model, which could further reduce the transmission overhead.

2.2 Architecture

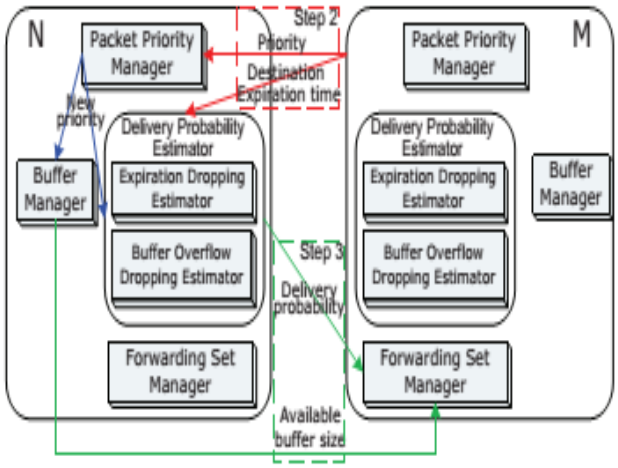

Fig. 1. SSAR overview where node N meets node M. The dashed rectangles enclose the information exchanged in step 2 and step 3

Figure 1 shows the architecture of SSAR, which has the following four components.

Packet priority manager: It calculates a priority for each buffered packet based on the willingness between nodes that the packet has traversed. This priority of a packet measures the social importance of the packet to the node.

Buffer manager: It manages buffers based on packet priority: (i) packets with priority 0 will not be buffered; (ii) when buffer overflows, packets of low priority are dropped first. That is, a new incoming packet can preempt the buffer occupied by lower-priority packets. This policy exactly follows the philosophy of "design for user".

Delivery probability estimator: It estimates a node's "delivery probability" of a packet, which is used to quantify the node's forwarding capability for that packet. A node forwards the packet to the neighbor with a higher delivery probability.

Traditionally, the quality of a relay is measured solely based on its contact opportunity to the destination node. SSAR measures the delivery probability of a node based on both of its contact opportunity to the destination and its willingness to forward.

It is straightforward that a node with a low contact opportunity should not be a relay. Interestingly, a node with a high contact opportunity but low willingness should not be a relay either .

Suppose S has a packet m1 to send to D, and it successively meets A, C, and B. If only contact opportunity is considered, it will forward m1 to A. Unfortunately, A will drop m1 since it is unwilling to forward for S(the edge weight is0). SSAR will avoid such forwarding. Though C is willing to forward m1, its willingness is so low that m1 may suffer high risk of being dropped, so SSAR will avoid such forwarding. As a result, B is the optimal forwarder for m1in this scenario, since it has high willingness to forward and a high contact opportunity. Forwarding set manager After a node determines a set of packets that should be forwarded to a better relay, existing routing protocols greedily transmit them no matter the receiver has enough buffers to hold these packets or not [9]. Obviously, bandwidth will be wasted if the transmitted packets are dropped due to buffer overflow. To address this issue, the forwarding set manager decides which packets to transmit by solving an MKPAR formulation. It considers the buffer constraint and transmits the packets that are most effective for social selfishness and routing performance.

2.3 Design Requirements

The design requirements include

•*Distributed:* We require that a network authority responsible for the administration of the network is only required to be periodically available and consequently incapable of monitoring the operational minutiae of the network.

• *Robust:* We require a misbehavior detection scheme that could tolerate various forwarding failures caused by various network environments.

• *Scalability:* We require a scheme that works independent of the size and density of the network

2.4 Literature survey

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

III. NEW IMPLEMENTATION

String transformation has many applications in data mining, natural language processing, information retrieval, and bioinformatics. String transformation has been studied in different specific tasks such as database record matching, spelling error correction, query reformulation and synonym mining. The major difference between our work and the existing work is that we focus on enhancement of both accuracy and efficiency of string transformation.

3.1 Registration:

An Author(Owner) or User have to register first,then only he/she has to access the data base. In that any of the above mentioned person have to login,they should login by giving their emailid and password. Then if an user wants to check the spelling, they can check and correct it automatically.

String Transformation: Here we are techniques for searching the String 1)String Generation,2)String Transformation.

String Generation: It means we have generated 50,000 Strings in alphabetical order.From a to z like a,aa,…..z.

String Transformation: It means we have given the user with the benefit of String Generation as well as String alias .It will be useful for the user for example if the end user have typed "TKDE" its equal to "Transactions on Knowledge and Data Engineering".

String mining: The User has to download the string with its meanings also He/She can download its substrings and its reverse etc.Also check the given string which is present in the bunch of strings ,if its present the result will be "String Found" otherwise "String NotFound".

In particular, TA judges if node Nj is a misbehavior or not by triggering the Algorithm 1. In this algorithm, we introduce BasicDetection, which takes j,Stask,Sforward, [t1,t2],R,Das well as the routing requirements of a specific routing protocol R,Das the input, and output the detection.

Algorithm 1 The Basic misbehavior detection algorithm

1:procedure BASIC DETECTION((j,S_{task},$S_{forward}$,[t_1,t_2],R,D))
2: for Each m∈ S_{task} do
3: if m ∉ $S_{forward}$ and R!=0 then
4: return 1
5: else if m∈ $S_{forward}$ and $N_k(m) \not\subset R$ then
6: return 1
7:else if m ∈ $S_{forward}$ and $N_k(m) \subset R$ and $|N_k(m)| < D$ then
8: return 1
9: end if
10: end for
11: return 0
12: end procedure

result "1" to indicate that the target node is a misbehavior or "0" to indicate that it is an honest node. The proposed algorithm itself incurs a low checking overhead. However, to prevent malicious users from providing fake delegation /forwarding /contact evidences, TA should check the authenticity of each evidence by verifying the corresponding signatures, which introduce a high transmission and signature verification overhead. We will give a detailed cost analysis in Section 4.2. In the following section, inspired by the inspection game, we will propose a probabilistic misbehavior detection scheme to reduce the detection overhead without compromising the detection performance.

3.2 Forwarding Set Optimization

In this subsection, we solve the following problem: suppose a node *M* contacts *N*, and *M* has determined a candidate packet set *C* for which *N* has higher delivery probabilities. Since *N*'s buffer may be inadequate to accept all packets in *C*, and the contact duration may be too short to transmit all these packets, how to determine a subset of *C* to transmit and in what order?We follow two principles. First,*M* will not forward a packet to *N* if *N* does not have sufficient buffers for that packet. According to the buffer management rule, *N*'s available buffer size *Lm* for *m* is:

$$L_m = L_0 + \sum_{\{k/p_k<p\}} \ell_k \quad \text{.....1}$$

where *L*0 denotes *N*'s empty buffer size, *{k|pk < p}* denotes the packets in *N*'s buffer whose priority is smaller than that of *m* (*p*), and *lk* denotes the size of packet *k*. Second, *M* tries to maximize its selfish gain through this contact, which is defined as follows.

Definition 1 (Selfish Gain) The selfish gain *g* that *M* achieves by forwarding *m* to *N* is the product of *m*'s priority *p* in *M* and the increment of delivery probability, i.e., $g = p \cdot \Delta P_{delivery}$.

Both factors in the definition are related to selfishness. *P* means how socially important the packet is. The larger *p* is, the more selfishness is gained. $\Delta P_{delivery}$ means how much this forwarding can increase the packet's probability to be delivered. The larger $\Delta P_{delivery}$ is, the more help is provided. So their product is a natural representation of the gained selfishness. Suppose all the packets in *C* are sorted by priority in the increasing order, then we can simply use *i* to denote the *ith* packet. Let *Xi* denote if packet *i* is selected by the to be transmitted subset (*Xi* = 1) or not (*Xi* = 0). According to the above two principles, the problem can be formulated as:

$$\max \sum_{i\in C} g_i X_i \quad \text{s.t.} \quad \forall i \sum_{j\le i} X_j \ell_j \le L_i \text{ ..2}$$

Next we convert it into an MKPAR formulation [10], where each item can only be assigned to a subset of the knapsacks. Suppose the original buffer is divided into |*C*| + 1 knapsacks such that the first knapsack has size S1 = *L*1, the *jth* (*j* ∈ 2, ..., |*C*|) one has size S*j* = *Lj* −*Lj*−1, and the (|*C*|+1)*th* one consists of buffers that cannot be preempted by any packet in *C*. Then packet *i* can only be packed into knapsacks indexed smaller than or equal to *i*. Let *Xij* denote if packet *i* is packed into knapsack *j* (*Xij* = 1) or not (*Xij* = 0), then *Xij* = 0 when *i* < *j*. Eq. 2 can be rewritten as an MKPAR:

$$\max \sum_{i=1}^{|c|} \sum_{j=1}^{|c|} g_i X_{ij}$$

$$\text{s.t. } \forall i \sum_{j} X_{ij} \le 1, \ \forall j \sum_{i} X_{ij} l_i \le S_j \text{ ...3}$$

Since a simpler variation of MKPAR has been proved by Dawande et al. [10] to be NP-hard, MKPAR is also NP-hard. Thus, we give a greedy algorithm, which ranks the packets in the decreasing order of selfish gain weighted by packet size, and packs them one by one until no more packets can be packed.

IV. PERFORMANCE EVALUATIONS

In this section, we evaluate the performance of SSAR and compare it to other existing routing algorithms.

4.1 Experiment Setup

The trace does not have the accurate social relationship information among participants, we need to construct a weighted directed social graph upon them. To better study SSAR, we evaluate it on two types of social graphs. The first type of social graph is probabilistically contact dependent. It can be built based on the following heuristic, which has been verified by many sociology studies [11]. The stronger tie two individuals have, the more likely they contact frequently. Individuals with more social ties are more likely to meet other people. Let $f*$ denote the overall contact frequency of the whole trace, fN denote node N's overall contact frequency, and fNM denote the contact frequency between N and M. The graph is constructed in four steps:

1) We generate power-law distributed node degrees based on several measurement studies [12].
2) We repeatedly assign those degrees to nodes in the trace, i.e., assign the largest degree to a node in such a way that node N's probability to be selected is $fN/f*$, and repeat this for the remaining degrees and nodes.
3) We generate weights for the social ties (edges) of each node. The best empirical data we can find about social tie strength is from one recent study in which participants rate their friendship nearly uniformly between 0 and 1 [13]. Thus, we generate weights for each node's social ties that are uniformly distributed within [0,1].
4) For each node N, we connect its ties to other nodes. We connect the strongest tie to another node in a way that node M's probability to be connected is fNM/fN, and repeat this for the other ties and not-connected nodes. In the end, for any ordered node pair $\overrightarrow{NM}$ that has not been connected yet, the weight of edge NM is set 0.

The second type is random social graph that is also constructed in four steps. The first and third steps are the same, but in the second and fourth steps we assign degrees to random nodes, assign weights to random social ties, and connect social ties to random nodes. One important feature of a social network is the average number of social ties per node; i.e., the number of nodes with a social tie strength larger than 0. In some networks, each node only has a few social ties; while in others, each node has many social ties. To generate social graphs with different average numbers of social ties per node, we fix the power law coefficient at 1.76 [12] when generating node degrees, but change the minimum acceptable degree.

4.2 Routing Algorithms and Metrics

*1) Rou*ting Algorithms*:* We compare SSAR with two other benchmark algorithms, PROPHET [2] and SimBet [3]. PROPHET is a standard non-oblivious benchmark that has been used to compare against several previous works [14]. It calculates a metric, delivery predictability, based on contact histories, and relays a packet to a node with higher delivery predictability. We use the same parameters as in [2], and age the delivery predictability upon every contact as done in [14].

SimBet has also been used as a benchmark in several works [15]. It calculates a simbet metric using two social measures (similarity and betweenness). A packet is forwarded to a node if that node has higher simbet metric than the current one. We use the same parameters as in [3]. Since the original algorithms do not define the order of packets to be transmitted during a contact, we adopt the transmission order used in RAPID [9]. Because this order has been shown to be the most effective, we believe such refinement does not favor SSAR in comparison. Since the original algorithm either assumes infinite buffer (SimBet) or assumes finite buffer but does not specify the packet dropping policy (PROPHET), we apply three policies (drop-tail, random drop, and minimum-utility-drop-first) in simulation, and only present the results of the best policy here, i.e., minimum utility- drop-first. Since it is impossible to traverse all dropping policies and choose the optimal one, we tried our best to impose the minimum influence on the original algorithms. PROPHET and SimBet are designed without considering social selfishness. For fair comparison, we modified them to be selfishness-aware. That is, nodes do not forward packets to others who are not willing to forward for them, and avoid immediate droppings caused by selfishness. However, when nodes forward packets to others who are willing to forward for them, they still follow the aforementioned transmission order and buffer policy. To show the effect of such selfishness awareness, we also include the basic PROPHET in our simulations. For convenience, we label the selfishness-aware PROPHET *PROPHET-1*, and label the basic one *PROPHET-2*.

2) Metrics: We use the following metrics to evaluate these algorithms: packet delivery ratio, the total number of transmissions, and selfishness satisfaction (SS). Packet delivery ratio is defined as the proportion of packets that are delivered to their destinations out of the total unique packets generated. The total number of transmissions can be used as a cost factor [3], and fewer transmissions mean lower cost. SS is defined as the ratio of the average priority of all forwarded or delivered packets over the average priority of all dropped packets. SS reflects how much users are satisfied with the network, because a larger SS indicates more important messages are served. SSAR, PROPHET, and SimBet are expected to have similar cost since all of them have only one replica of a packet.

V. THE ADVANCED ITRUST:A PROBABILISTIC MISBEHAVIOR DETECTION SCHEME IN DTNS

We start from Algorithm 2, which shows the details of the proposed probabilistic misbehavior detection scheme. For a particular node i, TA will launch an investigation at the

probability of pb. If i could pass the investigation by providing the corresponding evidences, TA will pay node i a compensation w; otherwise, I will receive a punishment C(lose its deposit). In the next subsection, we will model the above described algorithm as an Inspection Game. And we will demonstrate that, by setting an appropriate detection probability threshold, we could achieve a lower detection overhead and still stimulate the nodes to forward the packets for other node.

Algorithm 2 the proposed probabilistic misbehaviour Detection algorithm

---

1:initialize the number of nodes n
2:for i ⟵ 1 to n do
3:Generate a random number m_i from 0 to 10^n-1
4:if $m_i/10^n < p_b$ then
5:ask all the nodes (including node i) to provide evidence about node i
6:if BasicDetection(i ,$S_{task,}$ $S_{forward,}$[t1,t2],R,D) then
7: give a punishment C to node i
8:else
9: pay node i the compensation w
10: end if
11:else
12: pay node I the compensation w
13: end if
14:end for

The Reduction of Misbehavior Detection Cost by Probabilistic Verification

In this section, we give a formal analysis on the misbehaviour detection cost incurred by evidence transmission and verification. We model the movements and contacts as a stochastic process in DTNs, and the time interval *t* between two successive contacts of node *Ni* and *Nj* follows the exponential
distribution [17]:

$$p\{\leq x\} = 1- e^{-\lambda tj^{x}} ,x \in [0, \infty] \text{.....4}$$

where λij is the contact rate between *Ni* and *Nj* , the expected contact interval between *Ni* and *Nj* is $E[t] = 1\ \lambda ij$. We further denote *Costtransmission* as the evidences transmission cost and *Costverification* as the evidence signature verification cost for any contact. The below Theorem 2 gives a detailed analysis on the cost incurred by iTrust.

Theorem 2: Given that *pb* is the detection probability, $\bar{\lambda}$ is the mean value of all the λij , *T* is the inspection period, *N* is the number of nodes, *Costtransmission* and *Costverification* are the evidence transmission cost and evidence signature verification cost for a contact, the misbehavior detection cost in the whole network could be estimated as

$$\frac{1}{2} p_b \bar{\lambda} T \mid N \mid^2 *(Cost_{transmission} + Cost_{verification}).$$

….5

Proof: Given the above mentioned parameters, we could obtain the number of contacts |*H*| as

$$|H| = \frac{1}{2} \sum_i \sum_{j \neq i} T / \frac{1}{\lambda_{ij}} \approx \frac{1}{2} \bar{\lambda} T \mid N \mid^2$$

..6

If the detection probability is *pb*, the expectation of the transmission and verification cost for these contact evidences will be

$$E = p_b |H| = \frac{1}{2} p_b \bar{\lambda} T \mid N \mid^2 *(Cost_{transmission+} Cost_{vefication})$$

…..7

Equation 6 shows that between two time slots, the number of the contacts among |*N*| nodes is in line with the time *T* and the square of the number of the nodes. Then the cost of misbehavior detection (including evidence transmission and verification cost) is linear to the detection probability *pb*. From Theorem 2, it is observed that the misbehavior detection cost could be significantly reduced if choosing an appropriate detection probability without compromising the security level.

In the experiment section, we will show that a detection probability of 10% is efficient enough for misbehavior detection, which means the cost of misbehavior detection will be reduced to 10%, which will save a lot of resource of the TA and the network.

Exploiting Reputation System to Further Improve the Performance of iTrust

In the previous section, we have shown that the basic iTrust could assure the security of DTN routings at the reduced detection cost. However, the basic scheme assumes the same detection probability for each node, which may not be desirable in practice. Intuitively, an honest node could be detected with a low detection probability to further reduce the cost while a misbehaving node should be detected with a higher detection probability to prevent its future misbehavior. Therefore, in this section, we could combine iTrust with a reputation system which correlates the detection probability with nodes' reputation. The reputation system of iTrust could update node's reputation *r* based on the previous round of detection result, and, thereafter, the reputation of this node could be used to determine its inspection probability *p*. We define the inspection probability *p* to be the inverse function of reputation *r*. Note that, *p* must not be higher than the bound *g w+C* to assure the network security level, which has been discussed before. Further, it is obvious that *p* cannot be larger than 1, which is the upper bound of detection probability. If a node's *p* is 1, it means this node has been labeled as a malicious one and thus should be detected for all the time. What's more important, a node with a lower reputation will lead to a higher inspection probability as well as a decrease of its expected payoff πw.

VI. RESULT

1) Performance and Cost:The Effects of TTLWe change the packet TTL from 0 to 125 days to see its effects on the packet delivery ratio and the cost. Each node has 25 social ties.
Figure 2(a) shows the packet delivery ratio under the contact-dependent social graph. As the TTL increases, all algorithms can deliver more packets to the destinations. However, the delivery ratios will not increase after the TTL reaches a certain value, e.g., 25 days in SimBet and 50 days for the other three algorithms. This is because, after the TTL reaches some value, the forwarding capacity of the network becomes the performance bottleneck. Among all algorithms, SSAR has the highest packet delivery ratio

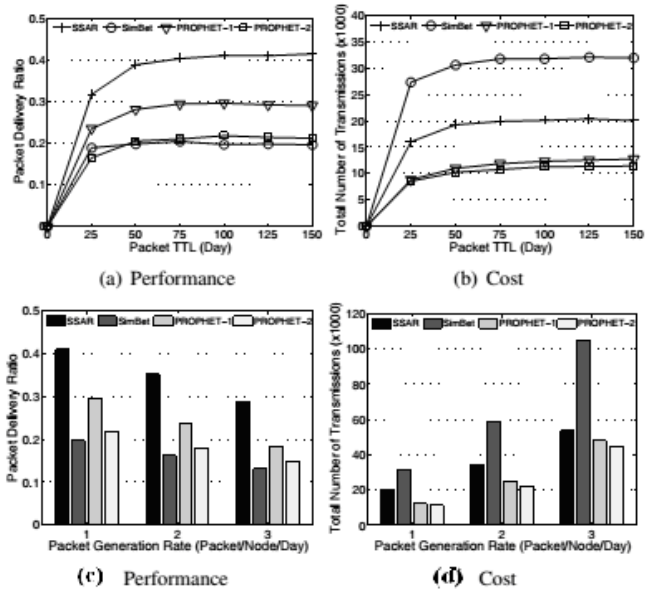

Fig. 2. Comparison of performance and cost. For (e), the packet TTL is 100 days. (a)(b) Results under the contact-dependent social graph. (c)(d) Results under different workloads on average.

Figure 2(b) shows the cost under the contact-dependent social graph. As the TTL increases, all algorithms have more transmissions, because packets stay longer in the network and have more opportunities to be transmitted. The total number of transmissions does not increase too much after the TTL reaches some value (about 75 days), when the number of transmissions is limited by the contact opportunities of the trace. Among the algorithms, SimBet has the most number of transmissions, which is about 60% more than that SSAR and 150% more than PROPHET-1 and PROPHET-2.

The Effects of Workload To evaluate the performance of SSAR under higher workloads, we change the packet generation rate from 1 packet per node per day to 3 packets per node per day. Each node on average has 25 social ties. Figure 2(c) and Figure 2(d) show the results. When the workload increases, all algorithms have lower packet delivery ratios and more transmissions. However, they change at different rates, especially SSAR and PROPHET-1. When the packet generation rate is 1 packet per node per day, SSAR delivers 40% more packets than PROPHET-1 with 60% more transmissions. When the rate increases to 3 packets per node per day, SSAR delivers 60% more packets than the latter with only 10% more transmissions. This means that SSAR is more efficient under high workloads.

The Effects of Workload

To evaluate the performance of SSAR under higher workloads, we change the packet generation rate from 1 packet per node per day to 3 packets per node per day. Each node on average has 25 social ties. Figure 2(c) and Figure 2(d) show the results. When the workload increases, all algorithms have lower packet delivery ratios and more transmissions. However, they change at different rates, especially SSAR and PROPHET-1. When the packet generation rate is 1 packet per node per day, SSAR delivers 40% more packets than PROPHET-1 with 60% more transmissions. When the rate increases to 3 packets per node per day, SSAR delivers 60% more packets than the latter with only 10% more transmissions. This means that SSAR is more efficient under high workloads.

The Effects of the Average Number of Social Ties perNode

The packet delivery ratio of SimBet and PROPHET-1 even drops a little bit. The reason is as follows. With the contact dependent social model, social ties are more likely to be added to nodes with frequent contacts first, and then to nodes with less frequent contacts. As a result, most later-added nodes contact each other less frequently, and the network's contact opportunity does not increase too much. Moreover, the extra data traffic due to the new social ties may overload the existing hot spots, affecting the packet delivery ratio negatively. Despite all these issues, SSAR still manages to deliver some more packets, because it makes more balanced use of social ties considering their contact opportunity, willingness, and buffer constraint.

The Effects of Willingness to Forward for Nodes without Social Ties

In previous simulations, a node's willingness to forward for others without social ties has been set as 0. In some networks, a generous user may be willing to forward packets for those who have no social tie with him, though the willingness is lower than that for those with social ties. To evaluate SSAR under such environments, we set a small weight for nodes without social ties to forward for others, and generate higher weights for nodes with social ties. In this case, each node can be seen as having a social tie with every other node, and PROPHET-2 becomes identical to PROPHET-1.

*2) **Allowed Selfishness**:* One key feature of SSAR is that it allows users to be socially selfish. To compare SSAR with other algorithms on how much selfishness is allowed, we plot the SS metric in Figure 4. The packet TTL is 25 days, and each node on average has 25 social ties. SSAR allows better selfishness than the other three algorithms. Specifically, SS in SSAR is one magnitude larger than that of the other three

algorithms. SSAR allows more selfishness because its buffer management policy satisfies social selfishness.

VII. DISCUSSION OF EXPERIMENT

The Impact of Various Packet Loss Rate on iTrust

In the previous section, we have shown that iTrust could also thwart the grey hole attack. In this section, we evaluate the performance of iTrust with different PLRs. In this experiment, we measure the scenarios of varying PLR from 100% to 80%. We set MNR as 10%, and the speed of 80 nodes varying from 10.5*m/s* to 11.5*m/s*. The message generation interval varies from 25*s* to 35*s*, and the TTL of each message is 300*s*. This implies iTrust will be effective for both black hole attack and grey hole attack. The misidentified rate is not affected by PLRs either. It is under 8% when the detection probability is under 10%. Thus the variation of PLR will not affect the performance of iTrust.

The Impact of Choosing Different Detection Probabilities

The above experiment results demonstrate that iTrust could achieve a good performance gain due to the following two reasons. Firstly, the detection performance of iTrust will not increase significantly as the increase of detection probability. Secondly, the inspection cost will increase along with the increase of the detection probability. Thus we suggest a lower detection probability such as 10% or 20%. And given the analysis of the inspection game, TA could set a proper punishment to ensure the detection probability. In this way, TA could thwart the misbehavior of the malicious nodes and stimulate the rational nodes.

VIII. CONCLUSION

In this paper, we have proposed a new statistical learning Approach to string transformation. Our method is novel and unique in its model, Routing algorithm, and misbehavior detection algorithm. Two specific applications are addressed with our method, namely spelling error correction of queries and query reformulation in web Search. Experimental results on two large data sets and Microsoft Speller Challenge show that our method improves upon the baselines in terms of accuracy and efficiency. Our method is particularly useful when the-problem occurs on a large scale.

REFERENCES

[1] R. Lu, X. Lin, H. Zhu, and X. Shen, "SPARK: A New VANET-based Smart Parking Scheme for Large Parking Lots", IEEE INFOCOM'09, April 2009.

[2]A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks",ACM SIGMOBILE CCR, vol. 7, no. 3, pp. 19–20, 2003.

[3]E.Daly and M. Haahr, "Social network analysis for routing in disconnected delay-tolerant MANETs, "Proc. MobiHoc, pp. 32–40, 2007.

[4] J. J. Jaramillo and R. Srikant, "Darwin: Distributed and adaptive Reputation mechanism for wireless ad-hoc networks" , Proc. MobiCom, 2007.

[5] H.Zhu, X.Lin, R.Lu, Y.Fan and X. Shen,"SMART:A Secure Multilayer Credit -Based Incentive Scheme for Delay-Tolerant Networks," IEEE vol.58,no.8,pp.828-836,2009.

[6] S.Reidt, M.Srivatsa, S.Balfe,"The Fable of the Bees: Incentivizing Robust Revocation Decision Making in Ad Hoc Networks" inCCS'09,2009.

[7] E. Ayday, H. Lee and F. Fekri,"Trust Management and Adversary Detection for Delay Tolerant Networks," inMilcom'10, 2010.

[8] B.B.Chen, M.C.Chan,"Mobicent:a Credit-Based Incentive System for Disruption Tolerant Network" IEEE INFOCOM'2010.

[9] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "Dtn routing as a resource allocation problem,"Proc. ACM SIGCOMM, 2007.

[10] M. Dawande, J. Kalagnanam, P. Keskinocak, R. Ravi, and F. Salman, "Approximation algorithms for the multiple knapsack problem with assignment restrictions,"Journal of Combinatorial Optimization,vol.4, pp. 171–186, 2000.

[11] M. Granovetter, "The strength of weak ties," The American Journal of Sociology, vol. 78, no. 6, 1973.

[12] A. Mislove, M. Marcon, K. P. Gummadi, Druschel, and Bhattacharjee, "Measurement and analysis of online social networks,"Proc. IMC, 2007.

[13] E. Gilbert and K. Karahalios, "Predicting tie strength with social media,"Proc. CHI, 2009.

[14] P. Hui, J. Crowcroft, and E. Yoneki, "Bubble rap: social-based forwarding in delay tolerant networks," Proc. MobiHoc, pp. 241–250, 2008.

[15] V. Erramilli, A. Chaintreau, M. Crovella, and C. Diot, "Delegation Forwarding,"Proc. MobiHoc, 2008.

