Available online at www.elixirpublishers.com (Elixir International Journal)

Computer Science and Engineering

Elixir Comp. Sci. & Engg. 38 (2011) 4473-4476

ISSN: 2229-712X

Enforcing socially selfish awareness routing among users

S.Lavanya and R.Ramya

Department of Computer Science and Engineering, Dr Pauls Engg College, Villupuram Dist.

ARTICLE INFO

Article history:

Received: 22 August 2011;

Received in revised form:

26 August 2011;

Accepted: 31 August 2011;

Keywords

Delay Tolerant Networks (DTNs),

SSAR,

MKPAR.

ABSTRACT

In the real world, most people are *socially selfish*; i.e., they are willing to forward packets with whom they have social ties but not to others, which varies with the strength of the social tie. A Social Selfishness Aware Routing (SSAR) algorithm to allow user selfishness and for having better routing performance in an efficient way is proposed. To select a forwarding node, SSAR considers both users' willingness to forward and their contact based approaches, which results in a better forwarding method than purely contact-opportunities. SSAR also formulates the packet forwarding process as a Multiple Knapsack Problem with Assignment Restrictions (MKPAR) to satisfy user demands for selfishness and performance. Trace-based simulations show that SSAR allows users to maintain selfishness and it achieves better routing performance with low transmission cost.

Introduction

The existing system, Delay Tolerant Networks (DTNs) [1] have the unique feature for the connectivity, which makes routing quite different from other wireless networks. Though many routing algorithms that have been proposed to increase data delivery reliability, they are clearly based on contact opportunity; i.e., without considering users' willingness and assuming that all nodes are willing to forward packets for others. In the real world, people are mostly selfish. In civilian DTNs such as PeopleNet [2] and Pocket Switched Network [3], a node may not be willing to forward packets for others. Then, previous algorithms may not work well since some packets are forwarded to nodes unwilling to relay, and will be dropped.

In this paper, we follow a new philosophy of, "design for user" which considers social selfishness as a user demand and allow socially selfish nodes to behave in the aforementioned ways to satisfy such demand. Thus we have to formulate the problem of how to enforce users' social selfishness in routing. This is not easy since the routing performance may be affected when social selfishness is considered.

We propose a Social Selfishness Aware Routing (SSAR) algorithm to address these challenges. To maintain social selfishness, SSAR allocates resources based on packet priority which is related to the social relationship among nodes. To maintain the routing performance, SSAR quantifies the relay's willingness to evaluateits forwarding capability. Moreover, SSAR formulates the forwarding process as a Multiple Knapsack Problem with Assignment Restrictions (MKPAR). It forwards the most effective packets for social selfishness and routing performance.

The following contributions are made,

Firstly, social selfishness into DTN routing is introduced.

A routing algorithm SSAR for DTNs, which follows the philosophy of design for user is presented.

A routing algorithm SSAR for DTNs, which follows the philosophy of design for user is incorporated.

Finally, the forwarding process as an MKPAR and provide a heuristic based solution is formulated.

Section II presents an overview of SSAR. Section III gives the detailed design. Section IV introduces the trace-driven simulations and discusses the results.

SSAR Overview

In this section, firstly our design philosophy is introduced, then discussion on models and assumptions, and finally an overview of SSAR is given and how it works is explained.

Philosophy: Design for User

The existing literature has focused on addressing individual selfishness such as using reputation-based, credit-based, or game-theory based approaches to stimulate users to cooperate and forward packets for others.

If the nodes cooperate with others, they will get help from others as a return; if not they will be punished. However, these incentive schemes may not be directly applied to deal with social selfishness, as the incentive schemes do not consider social selfishness.

By using incentives, every node will have to provide service to others without considering that there is a social tie or not. As a result, social selfishness is violated. We address this problem from a different point of view. We allow users to behave as what their social selfishness requires, but try to improve the routing performance under the social selfish behaviour.

Our underlying philosophy is that social selfishness is a kind of user demand that should be satisfied. It should be treated as a design metric to measure the user satisfaction. We call such design philosophy "design for user".

Models and Assumptions

Network Graph, the socially selfish network as a fully-connected weighted directed graph, where the vertex set V consists of all the nodes and the edge set E consists of the social links between nodes is designed.

The weight of edge A is A's willingness to forward packets for B. The weight of edge AB and that of BA may be different. The value of willingness is a real number within [0, 1], where 0 means unwilling to forward and 1 means the most willing to forward. The social willingness between two nodes depends on the social tie between them. The stronger the social tie is, the larger the social willingness is.

Tele:

E-mail addresses:

Fig. 1. SSAR overview where node N meets node M. The dashed rectangles enclose the information exchanged in step 2 and step 3 (Section II-D).

We assume that the willingness information is available. This should be achievable since each node only needs to know its willingness to forward for others. From the system perspective, this can be done as one system configuration step, in which the user assigns its willingness values for people he knows via some user interface in the mobile device. The user can set a default value (e.g., 0) for strangers. To be more user-friendly, the interface can provide several willingness levels such as "most", "much", "average", "poor", "none" for the user to choose from. The willingness information is configured when the user joins the network or migrates to a new mobile device, and is updated when his social ties change. However, such update is quite infrequent since social ties are usually stable.

Network Model In DTNs, nodes have limited bandwidth and computational capability. We assume each node has unlimited buffer for its own packets, but limited buffer for others. As for data traffic, we only consider unicast, and assume each packet has a certain lifetime (i.e., TTL). We further assume bidirectional links, which can be provided by some MAC layer protocols, e.g., IEEE 802.11.

Trust Model We assume the source of a packet is anonymous to intermediate nodes. For example, the source ID can be encrypted in a way so that only the destination can decrypt. Then intermediate nodes provide data forwarding service only based on the previous hop information. This assumption is not essential to SSAR, and we add it just to simplify the routing model. We also assume that some authentication service is available so that one node can not impersonate another. Otherwise, a node may claim to be someone else to obtain forwarding services from that node's social ties. How to provide such authentication service has been well studied Adversary Model In this paper, we only consider socially selfish behaviors.

Architecture

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

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

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. 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. Interestingly, a node with a high contact opportunity but low willingness should not be a relay either. This is illustrated in Figure 2(a). 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 is 0). 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 m1 in 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 [5]. 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.

The Protocol

We use Figure 1 to illustrate how SSAR works in the following five steps.

1) After neighbor discovery, node N and M deliver packets destined to each other in the decreasing order of priority. During packet delivery, they also exchange information related to their willingness to forward.

2) If N's willingness for M is positive, M sends N a summary list of _destination ID, expiration time, priority for its buffered packets.

3) From the priority information, N calculates the new priority value for each packet (Section III-A). Based on the new priority and other information in the summary list, N calculates its delivery probability (Section III-B) and available buffer size (Section III-C) for each packet in the list, and returns them to M.

4) M determines a candidate set of packets for which N has higher delivery probabilities.

5) Considering the available buffer size information, M further decides which candidates to transmit by solving the MKPAR (Section III-C) formulation. Packets will be deleted after being forwarded, so there is only one copy for each packet. Without loss of generality, in the last four steps we only describe how node M determines which packets to transfer to node N. Node N does so in similar ways. Though not very frequent in opportunistic DTNs, a node may be in contact with multiple neighbors at the same time. Then it would be very difficult to extend the MKPAR formulation to the whole neighborhood. As a simple solution, the node contacts neighbors one by one.

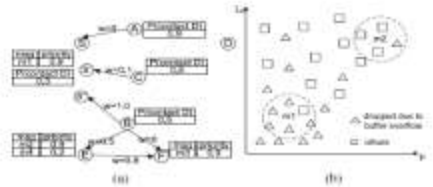

Fig. 2. (a) Examples of willingness-aware forwarding. (b) Heuristics used to estimate P_{meet}. Triangles and squares are historical samples.

Detailed Design

This section describes the detailed design of the packet priority calculation, the delivery probability estimation, and the forwarding set optimization.

Packet Priority

When a node receives and buffers a packet, it assigns a priority p to the packet. We borrow the idea of transitive trust [5] from the literature on reputation system and calculate packet priority in a chained way. Formally:

$$pi = pi-1 \cdot \omega i \geq 1 \quad (1)$$

where pi is the packet's priority in its ith hop and ω is the ith hop's willingness to forward the packets from the $(i-1)th$ hop. The initial priority $p0$ is set by the source. Since source anonymity is assumed, the packet source is not considered by intermediate hops. The priority assignment method and the buffer management policy are used to enforce social selfishness.

First, packets that traverse stronger social edges tend to have higher priorities. As shown in Figure 2(a), although *m*1, *m*2, and *m*3 have the same priority in the previous hop, they will receive different services in *B* after traversing different links. *m*3 will not receive any forwarding service; *m*1 will receive better service than *m*2. Second, packets from the same upstream node are also differentiated. In this example, *m*2 receives better service than *m*4 in *B* although they come from the same node. Since priority is updated hop by hop, it may improve cooperation in some cases. In Figure 2(a), if *m*3 from *F* arrives at *B* via *E*, its priority becomes 0.36 (instead of 0 through direct transmission) and will receive *B*'s service.

Delivery Probability Estimation

Suppose each packet has some expiration time, the question is: at a given time *t*, how to estimate node *N*'s probability of delivering packet *m* to its destination *D* before its expiration time *texp*?

Overall Delivery Probability

The overall delivery probability by *Pdelivery*. By definition the first and second dropping probability are given by $P\{texp \leq tc\}$ and $P\{tover \leq tc\}$, respectively. Note that the temporal order of *tc* and *texp* is determined by system parameters and the mobility pattern of *N* and *D*, while the time of buffer overflow depends on *N*'s traffic load. Thus we can assume that the two dropping events are independent. Then we integrate them to get the delivery probability:

$$Pdelivery = (1 - P\{texp \leq tc\})(1 - P\{tover \leq tc\}) \quad (2)$$

In DTNs with unpredictable connectivity, when *N* makes such estimation it is impossible to know the exact *tc*, and thus it is impossible to compute the r.h.s of Eq. 2. So we have to make some approximations. When *texp* > *tc*,

$P\{tover \leq tc\} \leq P\{tover \leq texp\}$ because the probability density function of *tover* is nonnegative. After inserting this inequation into Eq. 2, we get a conservative estimation:

$$Pdelivery \geq (1 - P\{texp \leq tc\})(1 - P\{tover \leq texp\}) \quad (3)$$

The above estimation of *Pdelivery* can be seen as determined by two independent droppings,

$P\{texp \leq tc\}$ and $P\{tover \leq texp\}$.

The first one means that the packet expires before *N*'s next contact with *D*, so we call it *expiration dropping probability* and denote it by *Pexp*. The second one means that the packet overflows before expiration, so we call it *buffer overflow dropping probability* and denote it by *Pover*. Next, we discuss how to estimate them individually.

Expiration Dropping Probability: To estimate *Pexp*, let random variable *X* denote the inter-contact time between *N* and the destination *D*. Assume that each inter-contact time is independent, then according to Markov's Inequality:

$$Pexp = P\{X > texp - \hat{t}\} \leq E(X)/(texp - \hat{t}) \quad (4)$$

where $E(X)$ is the mean of *X* and $\hat{t}$ is the most recent contact time between *N* and *D* before the estimation time *t*. $E(X)$ can be approximated by the average of historical inter-contact times. The value of *Pexp* should be bounded by 1. Eq. 4 intuitively means that nodes with a lower average inter-contact time (i.e., a higher contact frequency) with the destination have a lower expiration dropping probability.

Buffer Overflow Dropping Probability: The most important factor that affects *Pover* is *m*'s priority value *p* due to the buffer policy. Other two minor factors are the current empty buffer size *L*0 and the residual time $tr = texp - t$ before expiration. *L*0 is positively related to how long *m* can stay before being removed. But *tr* is negatively related: the longer *tr* is, the more likely it will be dropped due to buffer overflow. Whenever *N* drops or forwards a packet, it generates a record $< p, L0, tr, \beta >$. With data mining terminology, each record is called a *sample*, *p*, *L*0, and *tr* are called *feature dimensions* and β is called *class label*. $\beta = 1$ if *N* drops the packet due to buffer overflow and $\beta = 0$ if *N* does not drop it or drops it due to expiration. Our basic heuristic is that the probability that *m* will be dropped is similar to some historical packets which have similar feature values when they enter *N*'s buffer. Suppose we match *m* to a set *S* of similar packets, and its dropped subset is *Sdrop*, then *Pover* is estimated as:

$$Pover = |Sdrop|/|S| \quad (5)$$

Figure 2(b) illustrates the idea in a two-dimensional space $< p, L0 >$, where the historical packets in the dashed circle are the matched ones. In this example, the estimated *Pover* of *m*1 and *m*2 are 0.83 and 0.25, respectively. To match *m* to similar packets, we choose the K-Nearest- Neighbour (KNN) [6] algorithm from the data mining literature, which identifies the *K* packets that have the shortest distance to *m* in the feature space. However, KNN traverses all samples during matching, which induces high online computation cost, and leaves less contact duration time for data transmission. We combine KNN with the Kcenter algorithm [19] to propose a two-phase solution:

- In the offline phase (when not in contact with others), nodes use the K center algorithm to cluster samples into $\hat{K}$ clusters around $\hat{K}$ points in the feature space.
- In the online phase, nodes scan the $\hat{K}$ points in the increasing order of their distances with *m*'s feature vector until *K* samples are included in the scanned clusters.

Both the online and offline phase need to compute the distance between two feature vectors. When doing so, *L*0

$$D(m_1, m_2) = \sqrt{\sum_{i=1}^{3} \frac{\sigma_i^2}{\sigma_i^2} (m_1^i - m_2^i)^2} \quad (6)$$

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

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:

$$L_m = L_0 + \sum_{\{k | p_k < p\}} l_k \quad (7)$$

Second, *M* tries to maximize its selfish gain through this contact,

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.,

$$g = p \cdot \Delta Pdelivery.$$

According to the above two principles, the problem can be formulated as:

$$max \sum_{i \in C} g_i |X_i \quad s.t. \quad \forall i \sum_{j \leq i} X_j l_j \leq L_i \quad (8)$$

Let *Xij* denote if packet *i* is packed into knapsack *j* (*Xij* = 1) or not (*Xij* = 0), then *Xij* = 0 when *i* < *j*. Eq. 8 can be rewritten as an MKPAR:

$$max \quad \sum_{i=1}^{|\mathcal{C}|} \sum_{j=1}^{|\mathcal{C}|} g_i X_{ij}$$
$$s.t. \ \forall i \ \sum_{j} X_{ij} \leq 1, \quad \forall j \ \sum_{i} X_{ij} l_i \leq \mathbb{S}_j \qquad (9)$$

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.

Performance Evaluations

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

The evaluation is based on the MIT Reality trace [7] has validated the existence of the so-called "small-world" phenomenon, a well-known phenomenon in social networks.

The graph is constructed in four steps:

1) We generate power-law distributed node degrees based on several measurement studies [8].

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. is normalized to [0, 1] based on the total buffer size of *N*, and *tr* is normalized to [0, 1] based on the packet TTL.

Then their distance is:

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. 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 *NM* that has not been connected yet, the weight of edge

$\overrightarrow{NM}$ is set 0.

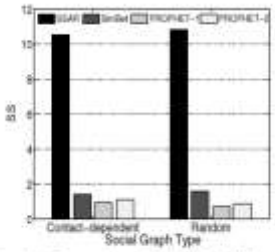

Fig. 4. Comparison of allowed selfishness.

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. Algorithms [8], [9] have also been proposed for finding the right relays for data forwarding in vehicular ad hoc networks. Recently, several algorithms [10], [11] use social metrics calculated from contacts. These approaches evaluate the forwarding capability of a node purely based on its contact opportunity. Individual selfishness has been widely studied in mobile ad hoc networks [4] and even in DTN [12].

Conclusion

In this paper, we introduce social selfishness problem and propose a routing algorithm SSAR following the philosophy of *design for user* SSAR allows user selfishness and improves performance by considering user willingness and contact opportunity. SSAR can maintain social selfishness and achieve a very good routing performance in an efficient way.

References

[1] K.Fall, "A delay-tolerant network architecture for challenged internets," Proc. SIGCOMM, pp. 27–34, 2003.

[2] M. Motani, V. Srinivasan, and P. Nuggehalli, "PeopleNet: engineering a wireless virtual social network," Proc. MobiCom, pp. 243–257, 2005.

[3] P. Hui, A. Chaintreau, J. Scott, R. Gass, J. Crowcroft, and C. Diot, "Pocket switched networks and human mobility in conference environments," SIGCOMM Workshops, 2005.

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

[5] A. Josang and S. Pope, "Semantic constraints for trust tansitivity," Proceedings of the Asia-Pacific Conference of Conceptual Modelling (APCCM) (Volume 43 of Conferences in Research and Practice in Information Technology), 2005.

[6] G. Mclachlan, Discriminant Analysis and Statistical Pattern Recognition. Wiley, 1992.

[7] A. Chaintreau, A. Mtibaa, L. Massoulie, and C. Diot, "The diameter of opportunistic mobile networks," Proc. ACM CoNEXT, 2007.

[8] J. Zhao and G. Cao, "Vadd: Vehicle-assisted data delivery in vehicular ad hoc networks," Proc. IEEE INFOCOM, 2006.

[9] Y. Zhang, J. Zhao, and G. Cao, "Roadcast: A popularity aware content sharing scheme in vanets," IEEE International Conference on Distributed Computing Systems (ICDCS), 2009.

[10] C. Boldrini, M. Conti, and A. Passarella, "Contentplace: Social-aware data dissemination in opportunistic networks," Proc. MSWiM, 2008.

[11] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in delay tolerant networks: A social network perspective," Proc. ACM MobiHoc, 2009.

[12] F. Li, A. Srinivasan, and J. Wu, "Thwarting blackhole attacks in distruption-tolerant networks using encounter tickets," Proc. IEEE INFOCOM, 2009.