Social-based Routing with Congestion Avoidance in Opportunistic Networks

Alexandru Asandei, Ciprian Dobre, and Matei Popovici

University Politehnica of Bucharest, Romania
Faculty of Automatic Control and Computers
alexandru.asandei@cti.pub.ro, ciprian.dobre@cs.pub.ro,
matei.popovici@cs.pub.ro

Abstract. In particular types of Delay-Tolerant Networks (DTN) such as Opportunistic Mobile Networks, node connectivity is transient. For this reason, traditional routing mechanisms are no longer suitable. New approaches use social relations between mobile users as a criterion for the routing process. We argue that in such an approach, nodes with high social popularity may become congested. We show that social-based routing algorithms such as Bubble Rap are prone to congestion, and introduce two algorithms *Outer* and *LowerEps*. We present experimental results showing that the latter outperform Bubble Rap and solve the congestion problem.

1 Introduction

The emergence and wide-spread of new-generation mobile devices together with the increased integration of wireless technologies such as Bluetooth and WiFi create the premises for new means of communication and interaction, challenge the traditional network architectures and are spawning an interest in alternative, ad-hoc networks such as *opportunistic mobile networks*.

An opportunistic mobile network (ON) [10] is established in environments where human-carried mobile devices act as network nodes and are able to exchange data while in proximity. Whenever a destination is not directly accessible, a source would opportunistically forward data to its neighbours. The latter act as carriers and relay the data until the destination is reached or the messages expire.

To cope with intermittent connectivity and ON partitioning, the natural approach is to extend the store-and-forward routing to store-carry-forward (SCF) routing [8]. In SCF routing, a next hop may not be immediately available for message forwarding. In this case forwarding will be delayed until a suitable node is encountered. Thus, ON nodes must be (i) capable of buffering data for a considerable duration, and (ii) selecting suitable carriers for a message, from the list of all sighted nodes. A bad forwarding decision may cause the packets to be delayed indefinitely [3].

Routing algorithms for ONs are either mobility-aware or social-aware. The former (which includes protocols such as PRoPHET [9]) takes routing decisions based on the number and duration of node encounters; the latter (which includes Bubble Rap [7]) relies on the knowledge that members (or nodes) of an ON are people carrying mobile devices. Social-aware ONs involve routing decisions based on how people are organized into communities, according to places of living and work, common interests, leisure activities, etc. A network of human relations as well as the overall structure of a community is captured by a social graph. Social networks are popular platforms for interaction, communication and collaboration between friends. The social relations between people can be generally inferred from the user interactions in such networks [12]. This is why in recent years, researchers have started to show an interest in social-based routing algorithms for ONs. However, approaches such as [7] can quickly lead to network congestions, as more popular ON members become flooded by message forwarding requests.

In this context, given that more powerful smarthpones and tablets continue to appear, would such devices be able to successfully carry messages destined for others, as envisioned by the opportunistic approach? Opportunistic networks are designed to support from catastrophic scenarios, where smartphones would take over the entire capacity of the damaged communication facilities in the disaster areas and beyond, all the way to completely distributed social networks, where mobile devices become caches of information in a publish/subscribe approach [10]. A forecast of mobile traffic published by CISCO [1] states that "*global mobile data traffic will increase 26-fold between 2010 and 2015. Mobile data traffic will grow at a compound annual growth rate (CAGR) of 92 percent from 2010 to 2015, reaching 6.3 exabytes per month by 2015*". Experts agree that at this growth a significant investment in the infrastructure will be needed. Opportunistic networks offer an alternative approach, when some of this wireless traffic can be offloaded directly into the mobile devices within the surrounding area. The current wireless communication infrastructure reached its threshold, as bottlenecks have already been observed - e.g., in [13] the overload of the AT&T infrastructure due to wider deployment of smartphones. Will this happen again when dispersing some of this overwhelming traffic directly to other smartphones? Unfortunately, predictions are not good, as they show that we still have to design communication protocols for opnets that can compensate for potential communication bottlenecks.

The contribution of this paper is twofold. First, we introduce the congestion problem, and show that the social-based Bubble Rap [7] algorithm can frequently produce message buffer overflows in nodes with high popularity, thus leading to lost messages. Second, we introduce two adaptations of the Bubble Rap algorithm, *Outer* which shows significant improvements in scenarios with a high number of ON participants and *Lower-Eps* which outperforms the classic Bubble Rap algorithm in situations where the number of ON nodes is low.

The rest of the paper is structured as follows: In Section 2, we review some of the most prominent social-aware routing algorithms, and especially Bubble Rap, which is considered to exhibit highest performance. Also, we show that approaches such as Bubble Rap are prone to congestion. In Section 3 we introduce two algorithms, *Outer* and *LowerEps*, and show that they behave better

```
begin BubbleProcedure()
    if (LabelOf(currentNode) == LabelOf(destination)) then
        if (LabelOf(EncounteredNode_i) == LabelOf(destination))
            and (LocalRankOf(EncounteredNode_i) > LocalRankOf(currentNode))
            and (message.InnerTokens > 0)
        then
            message.InnerTokens--
            EncounteredNode_i.addMessageToBuffer(message)
    else
        if ((LabelOf(EncounteredNode_i) == LabelOf(destination))
            or (GlobalRankOf(EncounteredNode_i) > GlobalRankOf(currentNode)))
            and (message.OuterTokens > 0)
        then
            message.OuterTokens--
            EncounteredNode_i.addMessageToBuffer(message)
end
```

Listing 1.1. The adjusted Bubble Rap procedure

than Bubble Rap, both in terms of performance, as well as susceptibility to congestions. In Section 4 we present our conclusions.

2 Congestion in ONs with social-based routing

2.1 Social-based routing algorithms

There is a considerable number of works regarding social routing in ONs as well as results showing the prevalence of these methods over traditional routing mechanisms [4].

In [15] a *Socio-Aware Overlay* is used for publish/subscribe communication. ON nodes act as: (i) publishers of certain events (abstracted as messages) and/or (ii) subscribers to certain events they are interested in. Also, there is a chosen event broker responsible for maintaining a high message delivery rate. Instead of relying on criteria such as geographical location for clustering ON nodes, the authors of [15] use communities to build an abstract topology of the ON. Community detection is achieved dynamically, as each node exchanges and updates community-related information (community affiliation, frequently encountered nodes, etc.) upon each encounter with another node. The HiBOp (History-based routing protocol for opportunistic networks) algorithm described in [4] also exploits social information for message routing. Unlike [15], where communities were inferred solely on the cumulative contact duration between nodes, HiBOp holds *context information* as a *set of attributes* which include personal information, device characteristics, residence, hobbies & fun, etc. for each node.

The Bubble Rap algorithm [7], builds on the ideas from [15], but unlike [4] assumes that no information regarding the characteristics of nodes is known in advance. In a manner similar to [15], Bubble Rap organises nodes into communities, and assigns a *local ranking* for each node n and each local community n is part of. Also, each node receives a *global ranking*. Both the local and global rankings are measures of importance (centrality) a node has either in some community, or in the entire ON. Routing is achieved as follows: messages will be forwarded to nodes with higher global ranking until a node from the destination's local community is encountered. Next, the same forwarding process occurs inside the local community, based on the local ranking instead of the global one. As is the case with [4], Bubble Rap shows significant improvements with respect to traditional (non-social) routing methods [7].

In most ON routing solutions, congestion is often ignored. For instance, in epidemic routing, which rely on broadcasting messages in parts of, or in the entire ON, authors of [11] argue that such protocols are able to ensure message delivery only when the *buffer capacity is sufficient*. In [5], Grossglauser and Tse propose a 2-hop forwarding approach which consider that *nodes have infinite buffer capacity*, which of course is unrealistic. Previous authors assume that nodes with higher popularity (either measured using community rankings or context similarity) are the key elements in routing messages, and are supporting most of the forwarding workload in the ON. Since Bubble Rap does not provide any mechanisms to control the number of copies for each forwarded message, it is hard to distinguish between situations where the entire network or just a subset of popular nodes is being flooded. In order to rule out the complete ON flooding scenario, we add some minor modifications to the original Bubble Rap algorithm which allows us to place an upper limit on the number of generated copies for each message.

```
begin Classic()
  foreach EncounteredNode_i do
    BubbleProcedure()
end
Obs: for each generated message, InnerTokens = OuterTokens = MAX_TOKENS/2
```

Listing 1.2. The Classic algorithm

We refer to the adjusted Bubble Rap algorithm as *Classic*. The logic implemented at the encounter of an arbitrary node from the ON is shown in Listing 1.1. The pseudocode for *Classic* is shown in Listing 1.2, and it relies on the *BubbleProcedure* from Listing 1.1. Whenever a message is generated, two values `InnerTokens` and `OuterTokens` expressing the message copy limit in the local and global communities, are associated with it. As seen in Listing 1.1, at lines 5 and 12, whenever the copy limit is reached, the message is no longer forwarded, and deleted from the node's buffer.

2.2 Experimental setting

We have tested *Classic* on a variety of traces with different features, in order to see whether congestion/saturation can occur. One of the most relevant features of each trace is the *interaction density*, i.e. the number of interactions between distinct users, per period of time. The traces used in the experiments can be roughly divided into three categories, based on interaction density: *low*, *medium* and *high*. *Low* density traces are typical for environments where any two wireless nodes (i.e., cars in traffic, large urban environments) meet with a low probability because of the large covered area. *Medium* density traces are typical for small communities (i.e. suburbs, small towns, university campuses), where node encounters have a higher probability than in case of *low* density scenarios. Finally, *high* density traces correspond to environments where node encounters are frequent (i.e. conferences, employees within a single company). Due to limited space, in this paper we have selected one representative trace for each category. All are publicly available traces, belonging to the CRAWDAD database:

- *Haggle-iMotte-infocom2005* (short *Infocom*): was collected at the IEEE Infocom Conference in Grand Hyatt Miami, USA. It has 41 participants and a duration of 3 days. We consider it to be of high interaction density.
- *Haggle-iMote-content* (short *Content*): resulted from an experiment performed in the city of Cambridge, UK, has 54 participants and lasted 8 weeks. It reflects mobility in a academic environment and also contains a number of 18 devices with a fixed location. We consider it to be of medium interaction density.
- *St. Andrews-sassy* (short *St.Andrews*): was collected in the city of St. Andrews, UK. It has 27 participants and lasted 11 weeks. It reflects mobility in an urban environment. We consider it to be of low interaction density.

The experiments aim at simulating an ON where nodes have the features extracted from a chosen trace. We follow the same assumptions from [14]: we consider an asynchronous communication pattern in which each node can generate a message for some other node(s) in the ON. Out of the total of N nodes in the network, $N/2$ nodes will be randomly chosen as generators of messages. Therefore, the number of messages each generator will send would be distributed according to the normal distribution. In this scenario, any of the $N-1$ nodes different from the generator has an equal chance of being a destination. However, we further enforce the observations from [14] and use a *community bias* b on the distribution of sent messages: all nodes that generate a number greater than b of messages, will use as destination only nodes from the **local** community. For all other nodes, any destination is generated at random. Such a scenario corresponds to the distribution of status messages in a social network as observed in [6].

Messages have an average size of 10KB. We assume each node has a buffer size of approximately 20MB, which means it can store up to 2048 messages. The choice of the buffer size is motivated by the current memory limit of most applications on Android devices, which ranges from 16 to 24 MB, on newer generation phones.

2.3 The congestion problem

We say a node becomes saturated whenever a *buffer overflow* event occurs for that particular node. The overflow is experienced when the buffer capacity is exceeded - for example, when a node receives messages at a higher rate than it is able to forward, thus causing message buffers to be filled. Whenever a new message is received and the buffer is full, it is discarded. We noticed a surprisingly large number of buffer overflow events during the lifespan of each trace, including those of low density, as can be seen in Figure 1. The results correspond to different message generation rates (nodes generate messages from every hour to as rare as every 5 days).

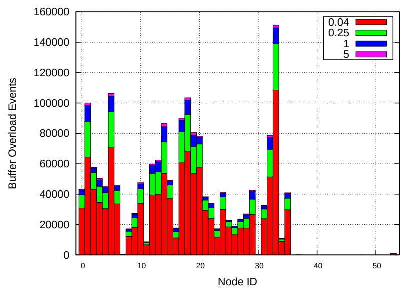
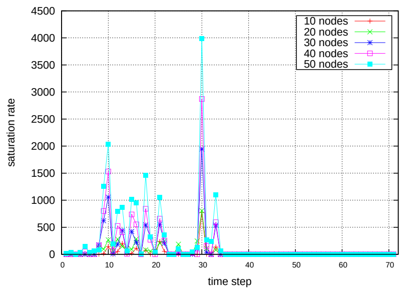


Fig. 1. Out-of-memory events, *Content*, 2 message copies

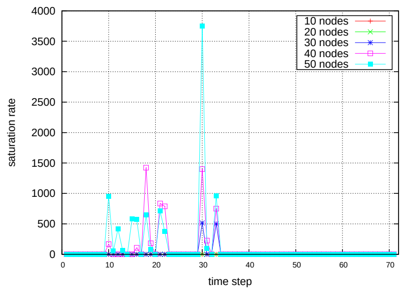
Next, we were interested to see how pervasive these events are. We therefore monitored the *saturation rate* for each node, i.e. the number of saturation events a node experiences, over a fixed period of time.

Figures 2(a) and 3(a) provide an overview of the saturation rate for the *Content* and *Infocom* traces. As can be noticed in Figure 3(b), even for traces of low density such as *St. Andrews*, high values for the saturation rate are recorded. The charts are plotted for the scenario when half of the total number of nodes generate new messages every 6 hours, which corresponds to a 0.25 message generation rate. The sample period for chart plotting is of 0.33, i.e. every 8 hours. Notice there is a phase-shift between between the moment when messages are generated, and the moment when congestion readings are performed. This allows messages to propagate in the network, and to avoid situations in which only the moments of message generation are recorded.

Finally, we looked at possible connections between saturation rates and the size of the ON. By using the same monitoring technique and running the simulations for subsets ranging from 10 to 50 nodes (or maximum available for each trace if lower than 50) we noticed that no such connection exists. Saturation does not depend on the size of the network, and can appear even in ONs of small size.

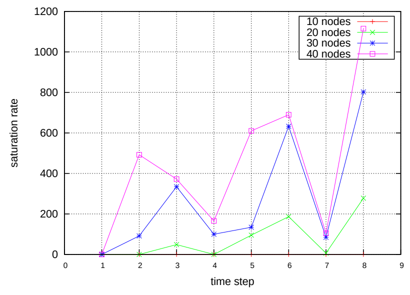


(a) Initial buffer size.

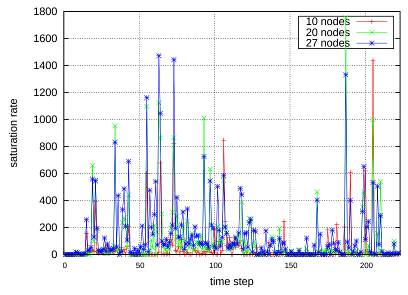


(b) 20 times increased buffer.

Fig. 2. Saturaton rate, *Content*, 2 message copies, 0.25 message generation period.



(a) Initial buffer size, *Infocom*.



(b) Initial buffer size, *St. Andrews*.

Fig. 3. Saturaton rate, 2 message copies, 0.25 message generation period.

Also, we were interested in seeing whether saturation could be avoided by an increase of the message buffer size. The simulations we carried out show that saturation still occurs even with buffer sizes increased up to 10 or 20 times over their initial values (depending on the data set used). This is shown in Figure 2(b), for the *Content* trace.

We therefore argue that node saturation is a *general problem* intrinsic to social-based routing algorithms (and in particularly to our adaptation of Bubble), and does not depend on specific features of the ON at hand.

3 Solutions for the congestion problem

3.1 The algorithms *Outer* and *LowerEps*

We examined two approaches for avoiding node saturation. The first one relies on the assumption that once a message has reached a node from it's destination community, it has a high delivery probability. Therefore, after a message copy has been forwarded inside the destination's community, all other copies will be reserved for encounters with nodes from other communities. This restriction is fit for relieving message flooding inside local communities. We denote by *Outer*, the adaptation of the *Classic* algorithm, that takes this into account. It can be seen in Listing 1.3. The restriction is implemented by setting `InnerTokens` to the value of 1, for each message.

Another important feature of *Outer* is related to forwarding messages that are not destined for the local community of the current node. While *Classic* would wait to encounter a node with a higher global ranking before sending the message, *Outer* sends the message to the first node encountered from another community, regardless of its rank. This has the effect of eliminating global rankings as a criterion for forwarding between nodes of different communities.

```
begin Outer()
  foreach EncounteredNode_i do
    BubbleProcedure()
    if (message.wasNotForwardedTo(EncounteredNode_i))
      and (LabelOf(currentNode) !=
        LabelOf(destination))
      and (LabelOf(currentNode) !=
        LabelOf(EncounteredNode_i))
      and (message.OuterTokens > 0)
    then
      message.OuterTokens--;
      EncounteredNode_i.addMessageToBuffer(message)
end
Obs: for each generated message, InnerTokens=1 and OuterTokens= MAX_TOKENS-1
```

Listing 1.3. The Outer algorithm

The second approach relies on the observation that, in *Classic*, messages having as destination nodes with low global ranking have to traverse nodes with higher ranking in the local community first, before being finally delivered.

In order to reduce the occurrence of such situations, we introduce another adaptation of *Classic*, namely *LowerEps*, that exploits randomness. The behaviour of *LowerEps* is governed by a threshold ϵ, as follows: for each node encounter there is a probability ϵ that a message will be forwarded to the encountered node, regardless of its rank, and a $1-\epsilon$ probability that the algorithm will behave precisely as *Classic*. Experiments show that *LowerEps* has an ideal behaviour for $\epsilon = 0.1$. The psedocode for *LowerEps* is shown in Listing 1.4.

```
begin LowerEps()
  BubbleProcedure()
  if  (message.wasNotForwardedTo(EncounteredNode_i))
    and (random < 0.1)
    and (message.Tokens > 0)
  then
    message.Tokens--;
    EncounteredNode_i.addMessageToBuffer(message)
end
Obs: for each generated message, InnerTokens = OuterTokens = MAX_TOKENS/2
```

Listing 1.4. The LowerEps algorithm

3.2 Experimental results

Our experimental results are aimed at: (i) examining the *performance* of *Outer* and *LowerEps*, with respect to *Classic* and (ii) studying the exposure to *congestion*, for each of the three algorithms. For the experiments we implemented a simulator that parses the traces and then applies an opportunistic routing algorithm at every encounter between two nodes.

For measuring performance, we use the following standard metrics: *hit rate*, i.e. the percentage of successfully delivered messages out of the total number of generated messages, *delivery cost* which represents the ratio between the total number of exchanged messages and the total number of generated messages and *latency* which is the time passed from a message being generated to it being successfully delivered to its destination.

To our knowledge, there is no widely accepted metric for measuring the susceptibility to congestions in ONs with social-based routing. For this reason, we consider *load balancing* to be a relevant criterion for such a task. Intuitively, an ON node i is less prone to congestion if the number of encounters of node i is proportionate to the number of messages sent to node i. This intuition is captured by the *load balancing factor* (LBF) for node i (denoted LBF(i)), which is the difference between the normalized sighting values for node i and those for the normalized number of received messages by node i, i.e.

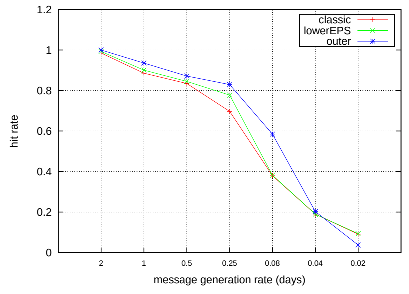
$$\mathrm{LBF(i)} = \frac{\mathrm{sight}(i)}{\max\limits_{i \in \mathrm{nodes(ON)}} \mathrm{sight}(i)} - \frac{\mathrm{msg}(i)}{\max\limits_{i \in \mathrm{nodes(ON)}} \mathrm{msg}(i)}$$

where nodes(ON) denotes the set of nodes in the network, sight(i) refers to the number of times node i was encountered by other nodes and msg(i) denotes the number of messages received by node i, during the lifespan of the trace.

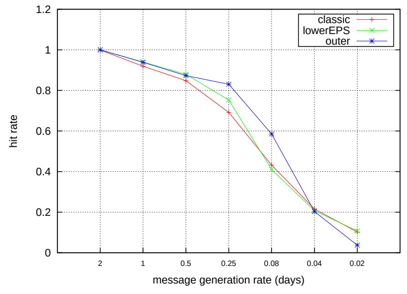
We use the average value of LBF(i) computed for all nodes i in the ON, to measure the exposure to congestion for each algorithm, on a given trace. We denote this value as *ALBF*.

Performance As can be observed in Figure 4(a), both *Outer* and *LowerEps* have a better average hit rate. The same is true when increasing the number of copies for each message, as can be seen in Figure 4(b).

The increase of the number of message copies, which is justified by the need of higher hit-rates, increases the delivery costs for *Outer*. However, when the number of generated messages is also increased, the delivery costs become moderate. The same thing is true about the average latency. Therefore, a suitable balance between (i) better hit-rate, and (ii) better latency and delivery costs can be found, depending on the expected traffic of the ON (i.e. number of exchanged messages).



(a) 2 message copies.



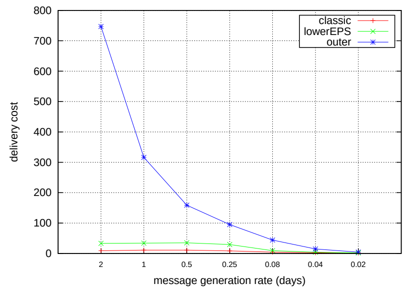
(b) 8 message copies.

Fig. 4. Hit rate, *Infocom*.

When dealing with a trace of a medium density such as *Content*, *Outer* achieves the best hit rate results.

This confirms the performance gain exhibited for high density traces. In terms of delivery cost and latency, the values for *Outer* are still higher but to a lower extent. This tells us that pushing messages towards outside communities works best when at least half of the total number of nodes belong to communities that interact with each other frequently. We have obtained similar results on other medium density traces, such as *UPB2012* [2].

A different picture presents itself when doing the same simulations for the low density trace of *St. Andrews*. The best performance in this case is achieved


(a) *Infocom*.

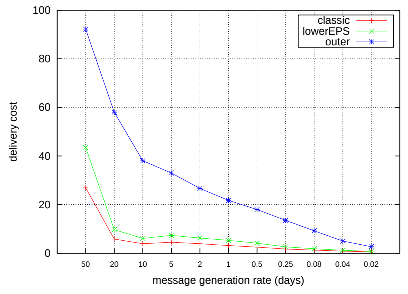
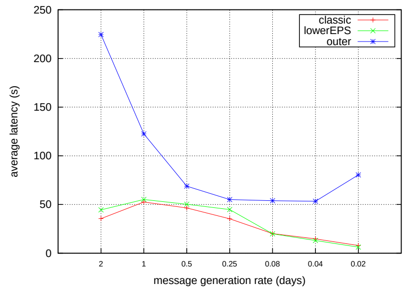

(b) *Content*.

Fig. 5. Delivery cost, 2 message copies.


(a) *Infocom*.

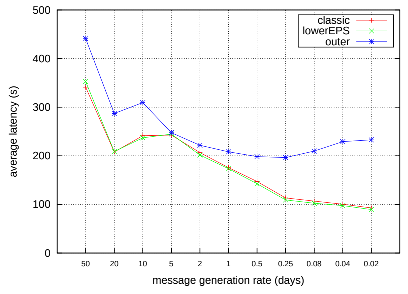
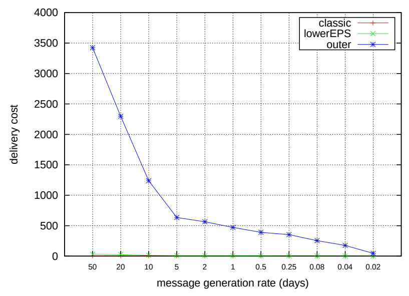

(b) *Content*.

Fig. 6. Average latency, 2 message copies.

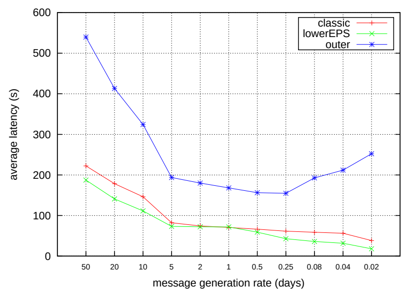
by *lowerEPS* which has better values for both hit rate and average latency and a delivery cost comparable to *Classic*. This shows that when faced with few interactions in a large geographical space, node centrality is not very reliable as there is not enough data for the computation of a proper value. In such a scenario, an approach which exploits randomness proves to be more useful. Figures 7(a) and 7(b) also show that the routing scheme used by *Outer* is inappropriate for this scenario, having the lowest hit rate. Similar results have been obtained for the low-density trace *UPB2011* [2].

Exposure to congestion In opportunistic networks, *load balancing* depends on the *sighting distribution*. By *sighting distribution* we denote the number of times each node has been sighted (by any other node) during the life span of a trace and is equivalent to the number of interactions. The more times a node interacts with other nodes, the more likely it is to receive messages. Thus, perfect load balancing would result from the use of a totally random routing scheme. Ideally, the distribution of received messages for each node of the ON would be almost identical to the sighting distribution. Under this assumption, we plot the *LBF* value for *Classic*, *Outer* and *LowerEps*, against that of a totally random routing scheme.

Figures 8(a), 8(b) show the result of applying this technique, for the three traces. Due to the nature of opportunistic networks, a critical mass of messages



(a) Delivery cost.

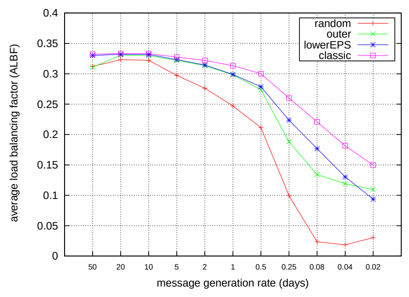


(b) Average latency.

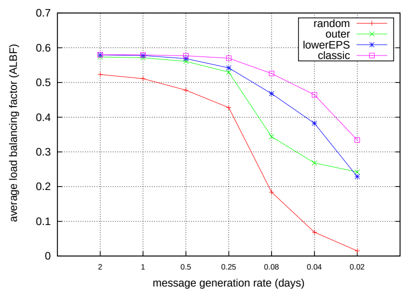
Fig. 7. *St. Andrews*, 2 message copies.

(thus a lower message generation period) in the network is required in order to achieve a close to perfect load balancing even with a random routing scheme.

Figures 8(a) and 8(b) account for the performance of *Outer*. They show that *Outer* achieves the best load balancing, being closest to the random routing algorithm. It is notable that *Classic* has the worst behaviour, which shows that it is unfit to provide good load balancing.



(a) *Content*.



(b) *Infocom*.

Fig. 8. ALBF, 2 message copies.

4 Conclusions

We have shown that congestions are likely to appear in social-based routing algorithms due to buffer overflows, an issue which has been, to our knowledge, insufficiently addressed. We have identified two solutions to the congestion problem. *Classic*, *Outer* and *LowerEps* use different strategies for forwarding messages not destined for the local community of the current node. The performance results suggest that global rankings are not always a suitable criterion for forwarding messages between nodes of different communities. All these observations were drawn from experiments performed on a variety of traces, with different number of nodes, taken in various environments and having a broad range of sighting distributions.

Acknowledgment

The research presented in this paper is supported by project "ERRIC - Empowering Romanian Research on Intelligent Information Technologies/FP7-REGPOT-2010-1", ID: 264207.

References

1. Cisco visual networking index: Global mobile data traffic forecast update, 20112016. http://www.cisco.com/.
2. Alexandru Asandei. Socio-aware routing in opportunistic networks. In *Bachelor Thesis*.
3. Fredrik Bjurefors, Per Gunningberg, and et al. Congestion avoidance in a data-centric opportunistic network. In *Proc. of the ACM SIGCOMM workshop on Information-centric networking*, ICN '11, pages 32–37, New York, USA, 2011.
4. Chiara Boldrini, Marco Conti, and Andrea Passarella. Exploiting users and social relations to forward data in opportunistic networks: The hibop solution. *Pervasive and Mobile Computing*, 4(5):633 – 657, 2008.
5. Matthias Grossglauser and David N. C. Tse. Mobility increases the capacity of ad hoc wireless networks. *IEEE/ACM Trans. Netw.*, 10(4):477–486, August 2002.
6. Bernardo A. Huberman, Daniel M. Romero, and Fang Wu. Social networks that matter: Twitter under the microscope. `http://ssrn.com/abstract=1313405`, December 5 2008. CoRR, vol. abs/0812.1045.
7. Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble rap: social-based forwarding in delay tolerant networks. In *Proc. of the 9th ACM int. symp. on Mobile ad hoc net. and comp.*, MobiHoc '08, pages 241–250, New York, NY, USA, 2008. ACM.
8. Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerant network. In *Proc. of the 2004 conf. on Applications, technologies, architectures, and protocols for computer comm.*, SIGCOMM '04, pages 145–158, New York, USA, 2004. ACM.
9. Anders Lindgren, Avri Doria, and Olov Schelén. Probabilistic routing in intermittently connected networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):19–20, July 2003.
10. Luciana Pelusi, Andrea Passarella, and Marco Conti. Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Comm. Magazine*, 44(11):134–141, November 2006.
11. Amin Vahdat and David Becker. Epidemic routing for partially connected ad hoc networks. Tech. Rep. CS-2000006, Duke University, Durham, NC, December 2000.
12. Christo Wilson, Bryce Boe, and et al. User interactions in social networks and their implications. In *Proc. of the 4th ACM Eu. conf. on Computer systems*, EuroSys '09, pages 205–218, New York, USA, 2009. ACM.
13. Jenna Wortham. Customers Angered as iPhones Overload AT&T. `http://www.nytimes.com/2009/09/03/technology`, September 2 2009. The New York Times, [Online; accessed 2-November-2012].
14. Kuang Xu, Pan Hui, and et al. Impact of altruism on opportunistic communications. In *Proc. of the first int. conf. on Ubiquitous and future networks*, ICUFN'09, pages 153–158, Piscataway, NJ, USA, 2009. IEEE Press.
15. Eiko Yoneki, Pan Hui, ShuYan Chan, and Jon Crowcroft. A socio-aware overlay for publish/subscribe communication in delay tolerant networks. In *Proc. of the 10th ACM Symp. on Modeling, analysis, and simulation of wireless and mobile systems*, MSWiM '07, pages 225–234, New York, NY, USA, 2007. ACM.