Searching a Needle in (Linear) Opportunistic Networks

Esa Hyytiä
Aalto University
Finland
esa@netlab.tkk.fi

Suzan Bayhan
University of Helsinki
Finland
bayhan@hiit.fi

Jörg Ott
Aalto University
Finland
jo@netlab.tkk.fi

Jussi Kangasharju
University of Helsinki
Finland
jakangas@cs.helsinki.fi

ABSTRACT

Searching content in mobile opportunistic networks is a difficult problem due to the dynamically changing topology and intermittent connections. Moreover, due to the lack of global view of the network, it is arduous to determine whether the best response is discovered or search should be spread to other nodes. A node that has received a search query has to take two decisions: (i) whether to continue the search further or stop it at the current node (current search *depth*) and, independently of that, (ii) whether to send a response back or not. As each transmission and extra hop costs in terms of energy, bandwidth and time, a balance between the expected value of the response and the costs incurred must be sought. In order to better understand this inherent trade-off, we assume a simplified setting where both the query and response follow the same path. We formulate the problem of optimal search for the following two cases: a node holds (i) exactly matching content with some probability, and (ii) some content partially matching the query. We design *static search* in which the search depth is set at query initiation, *dynamic search* in which search depth is determined locally during query forwarding, and *learning dynamic search* which leverages the observations to estimate suitability of content for the query. Additionally, we show how unreliable response paths affect the optimal search depth and the corresponding search performance. Finally, we investigate the principal factors affecting the optimal search strategy.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design studies; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design — *Wireless communication*

Keywords

Mobile opportunistic networks, mobile search, mobile cloud computing, dynamic programming.

MSWiM'14, September 21–26, 2014, Montreal, QC, Canada.

ACM 978-1-4503-3030-5/14/09 ...$15.00.
http://dx.doi.org/10.1145/2641798.2641828 .

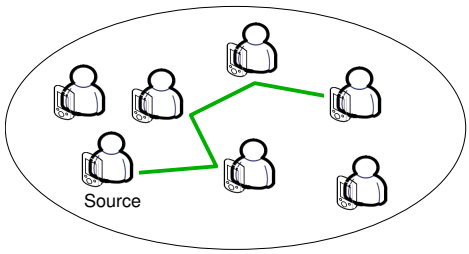

Figure 1: Search query travels from a node to another and the path forms a linear trajectory in space. The response is assumed to follow the same path backwards.

1. INTRODUCTION

Mobile opportunistic networks, also dubbed *Pocket switched networks* [6, 10], are networks in which mobile devices carried by humans can exchange information via short-range communication interfaces, e.g., Wi-Fi, Bluetooth, when they come in transmission range of each other and physically carry the content on their way. Certainly, this operation mode is vital for cases where the network infrastructure fails (e.g. after natural disasters), does not exist, or access to infrastructure services or even the Internet at large is blocked [3]. In addition, this communication involves only the peers in wireless contact, in contrast to the frequently used cloud-based third party services (e.g. Dropbox) which may take a long distance detour across the Internet and potentially half-way around the world. *Delay-tolerant networking* (DTN) [4, 5] defines such a networking paradigm facilitating communication without an infrastructure support for a variety of application scenarios including inter-planetary, vehicular, underwater, and opportunistic networks.

The wealth of data produced or downloaded by the mobile devices requires efficient search algorithms that can locate the relevant content quickly and cost-effectively rather than naï¿½ve enquiry of each node upon a contact. Searching content in mobile opportunistic networks is a difficult problem due to the dynamically changing topology and intermittent connections. A question arising in this context is what are the fundamental determinants of search in mobile opportunistic networks. In this work, we aim to provide insights on this question by designing static and dynamic search schemes. We focus on a single query that visits a node after another along some (natural) path as illustrated in Fig. 1 (i.e., the query is not replicated). The response follows the same path backwards. However, the response

path is assumed to be unreliable, e.g., due to mobility during the search. More specifically, we assume that searches terminate relatively quickly (say, order of ten seconds) and a link backwards exists if the response can be transmitted shortly (but not necessarily immediately). In other words, we do not require persistent end-to-end paths.

In practice, the search path can form naturally based on some path selection criteria such as a similarity metric for nodes, which reflects positively to the probability of finding relevant information in the node. Similarly, the actual search can consist of multiple (independent) linear paths.

In this paper, we consider three types of search strategies: static, dynamic, and learning dynamic search. Regarding a node's value for the issued query, first we assume that a node either holds a relevant content or not for the query. Second, we assume that a node may hold some relevant content whose relevance has either uniform or exponential distribution. The former corresponds to the *binary response* whereas the latter is the *partial response* scenario. We start with *static strategies* where the search is extended to a predefined number of nodes n. Then we consider *dynamic strategies* that may stop the search before depth n depending on what has been found so far (and whether some response has already been sent back). Both of these assume that each node knows the distribution of information in the nodes (value of response to given search). Our final search strategy, referred to as the *learning strategy*, is more robust and estimates the value distribution dynamically as the search progresses from a node to another. Although we assume that response follows the same path as the query, we model the unreliability of the link between two nodes on the response path and analyze how it affects the optimal search. The results in this paper serve two purposes: First, they enable design of efficient distributed search algorithms. Second, they provide insight on under what circumstances searching for content in mobile opportunistic networks is feasible and present the optimal operating region (in terms of search depth) for various scenarios.

Rest of this paper is organized as follows. First, in Section 2 we briefly review the related work. Then, Section 3 introduces our model and notation. The different search strategies are analyzed in Section 4, followed by a performance evaluation in Section 5, and a discussion in Section 6. Section 7 concludes the paper.

2. RELATED WORK

In a broad context, we can consider every forwarding algorithm in a DTN as a search scheme for a specific target node. We exclude broadcast algorithms as they aim to reach each and every node. In content search, first the searched content is mapped to some node(s) that have a high likelihood of holding this particular content. Next, nodes upon encounters forward the query with the aim of reaching the specified destination(s) that matches the mapping between content and the node profile. For example, *seeker-assisted search* (SAS) [2] vaguely maps a content to the nodes of a particular community which is a group of nodes sharing common interests. Hui et al. [6], design Haggle – a content sharing scheme, by leveraging the node's self-declared interests to locate the contents that might fall in the interest of the node. In Haggle, each content and node have some attributes that are manually defined. These attributes provide the basis of mapping between a content and its target nodes. See also the *Bubble* forwarding algorithm [7], which tries to exploit the social structures when making the forwarding decisions.

Any rational search scheme should direct the search towards the nodes that have higher likelihood of holding the searched item. This forwarding decision can exploit various characteristics of the network, e.g., the community structure, content and node relevance. For example, SAS [2] exploits the *homophily principle*, tendency to associate and interact with similar others, and directs the search towards the nodes of the same community as this content might have been searched and be readily available at a node in this community. SAS expands the search to the other parts of the network, although there is a lower probability of matching there, to avoid searching only in a particular part of the network. In Haggle, nodes exchange contents at each encounter so that contents are constantly *pushed* towards the nodes with some interests for this content rather than an explicit search. In this paper, similar to SAS, we consider a *pull-based* search scheme in which a node issues a query for finding a specific content.

Another challenge in opportunistic search is deciding when to stop the spreading of the search query. As nodes operate distributedly, the completion of search cannot be signalled immediately to the other parts of the network. Hence, each node should decide on forwarding or terminating the spread. An early termination may result in search getting no responses whereas late termination leads to over-consumption of the resources, e.g., battery. Pitkänen et al. [9] define a termination logic in which each node using the observed degree of itself estimates the number of nodes the query might have been received by and the number of possible responses generated by these nodes. The query is terminated if the estimate is above some threshold. Under transmission bandwidth and storage capacity constraints, RAPID [1] replicates the messages to the node's contact in decreasing order of message utilities such as expected delivery delay and deadline violation level. In this manner, messages yielding higher cost compared to their utilities are terminated based on the benefit and cost evaluation at each node. Setting time-to-live (TTL) for a query is another way of limiting the spread as a message is dropped after the expiry of its TTL. However, determining the optimal TTL is not straightforward as it depends on various network dynamics including the traffic load and content availability. Our solution is similar to [1] in the sense that each node evaluates the expected utility of the next hop and the increased cost due to involving it. This decision can be intricate depending on the degree of information available to the decision maker. In this work, different than the listed approaches, we find the *optimal depth* - the hop distance from the searching node - to stop the search under various settings.

Search, although having similarities with opportunistic forwarding, is more complicated due to its birectional nature, i.e., the discovered content or other responses have to be forwarded back to the searching node. What is more, treating search as a twofold process, e.g. *query forwarding* and *response forwarding*, may lead to a sub-optimal performance or even hinder the search success. For instance, search message eventually discovering some related content, might already be too far from the searching node that the response is obsolete or too difficult to route back. Therefore, the response path should also be taken into account

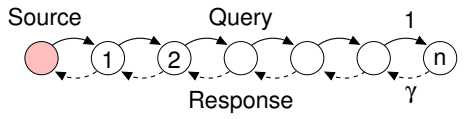

Figure 2: Linear network, where a query travels to the right and a possible response(s) to the left.

explicitly. In this work, we assume that the response messages follow the path of the query backwards. In our basic model, we assume that this path exists for the duration of the search, but numerically we also investigate what happens if the path back to the searching node becomes unreliable.

3. MODEL AND NOTATION

As already mentioned, searching content in an opportunistic wireless network is not trivial. Therefore, we resort to analyze a simplified setting to understand how much an optimal search scheme can save. In particular, we consider a search in a linear network, where the basic action at each node is to decide if the search should continue further, or if we are satisfied with the content found so far. More specifically, our model is as follows.

- We assume a linear network, where the source node is located at the origin and there are an infinite number of nodes along the positive x-axis, see Fig. 2. Please note that this linear model is a logical abstraction rather than a physical interpretation (cf. Fig. 1), however under our assumption of no replication for query messages, every query will follow such a linear path.
- A query travels on the *forward path*, where the loss probability is assumed to be zero. (A node meets another node within a reasonable time with a high probability).
- A possible response travels in the opposite direction on the *backward path* and the response can be delivered to the previous node in the chain with a fixed probability of γ (during the search). In the ideal case, $\gamma = 1$. However, in practice, depending on the connectedness of the network, there are many reasons for $\gamma < 1$, including that a node may have carried the search request away or that a previous hop may have gotten out of range.
- For each query, each node i has a response which value is described by i.i.d. random variables denoted by V_i. Note that $V_i = 0$ corresponds to "nothing useful". This value can be interpreted as the ranking or relevance of the response similar to ranked search results returned by a search engine.
- We let m denote the total number of transmissions when the search has completed, and d the highest valued response that is returned back to the source. A possibly failed transmission on the return path is also included in m.
- Each transmission costs e (say energy & time), which is assumed to be the same for both the query and response for simplicity. In practice, responses might have a higher cost if they return a lot of data (e.g., music, photos, ...)
- If a search is terminated after n hops and nothing useful has been found, there is no need to send a response back to the source, $n = m$, and the transmission costs are ne.
- As the metric we consider the net profit of a search, i.e., *the utility*, which is the value of the response minus the expenses,

$$U := d - m \cdot e. \tag{1}$$

- Node i along the search path must choose its action from the following four options:
  1. Stop the search
  2. Stop the search and send a response back to the source
  3. Continue the search to Node $i + 1$
  4. Continue the search to Node $i + 1$ and also send a response back to the source
- The optimal search algorithm α chooses dynamically the action that maximizes the utility given by (1).

We note that our problem is related to the *optimal stopping problem* in the routing at DTNs, see, e.g., [8] and [12]. However, there is a fundamental difference because in our setting there are multiple ways to "stop": one can simply stop and give up, or stop and send a response back to the searching node (which costs more in terms of energy and time). Moreover, it is possible to send a response while still proceeding further with the search. The search forwarding algorithm must take all these different options and the earlier observations into account when making the decisions.

4. OPTIMAL SEARCH STRATEGIES

In this section, we will analyze the optimal search strategies. We consider static strategies, where the search distance is defined at the start, and dynamic strategies, where the actions may depend on what has been found so far, and if some responses have already been sent. The dynamic strategies may also learn the value distribution during the search.

4.1 Bernoulli distribution

Let us start with the binary case where a node either has the complete response to the query, or no relevant information at all. That is, the value of the response from node i obeys Bernoulli distribution $V_i \sim \text{Bernoulli}(p)$, where p denotes the probability that a node has the searched content. We refer to p as the *content availability*, and q denotes the probability of the opposite case, $q = 1 - p$. Moreover, the links on the return path can be unreliable ($\gamma < 1$), and at most one response is sent per query.

4.1.1 Static strategy

A static search strategy is defined by a fixed depth n, i.e., each search will check the first n nodes and then return the highest response found. The number of transmissions on the return path is $r_n = \sum_{i=1}^{n} i\gamma^{i-1}(1-\gamma) + n\gamma^n$, which gives

$$r_n = \begin{cases} \dfrac{1-\gamma^n}{1-\gamma}, & \text{when } 0 < \gamma < 1, \\ n, & \text{when } \gamma = 1. \end{cases} \tag{2}$$

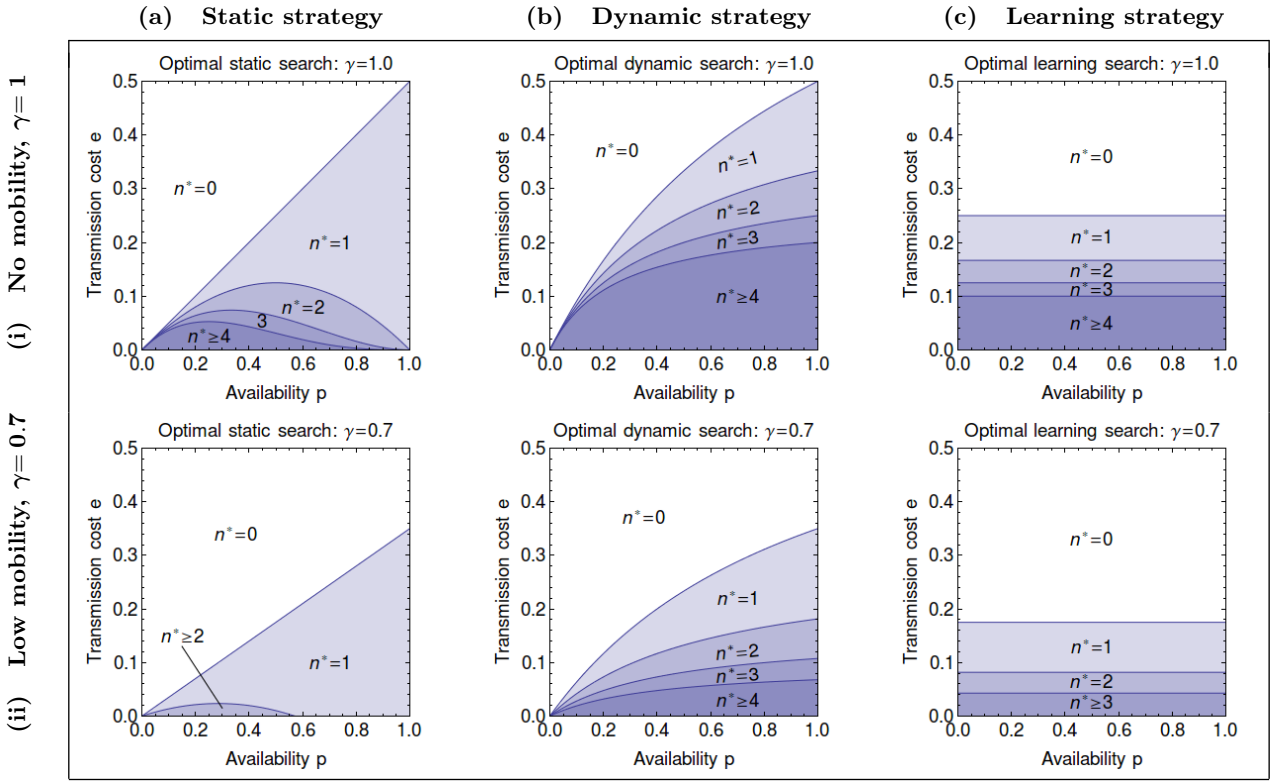

Figure 3: Optimal max. search depth n^* in Bernoulli case ($\mathbf{v}_{\max} = 1$) with (a) the static strategy, (b) dynamic strategy and (c) learning strategy (which does not know p *a priori*). The top row corresponds to the ideal case with $\gamma = 1$, whereas on the bottom row the return path is unreliable and $\gamma = 0.7$.

The total number of transmissions is $m = n + r_n$, and the response reaches the searching node with probability of γ^n. The expected search result with depth n is

$$\begin{aligned} R_n &= \mathrm{E}[\max\{V_1, .., V_n\}] \cdot \gamma^n \\ &= (0 \cdot q^n + 1 \cdot (1 - q^n))\, \gamma^n \\ &= (1 - q^n)\gamma^n. \end{aligned} \tag{3}$$

Thus, the expected utility under n hop search is

$$U_n = R_n - (n + r_n)e.$$

which reduces to

$$U_n = (1 - q^n)\gamma^n - (n + r_n)e, \tag{4}$$

and with $\gamma = 1$,

$$U_n = 1 - q^n - 2ne. \tag{5}$$

The optimal static policy is obtained by finding the depth n that maximizes the expected utility:

$$n^* = \arg\max_{n \in \mathbb{N}} \quad U_n. \tag{6}$$

We note that this is clearly a non-optimal strategy: if Node 1 already has the searched content it is useless to search any further. Nonetheless, we consider this simple strategy first and later compare how far it is from the optimal.

Case $\gamma = 1$.

Let us first assume the ideal case with $\gamma = 1$. Note that if $p < 2e$, then the optimal search depth n is zero, i.e., it is not worth initiating a search at all. The optimal (integer-valued) search depth n is found by studying the gain from expanding the search by one step, i.e, $\Delta U(n) = U_{n+1} - U_n$:

$$\Delta U(n) = \big(1 - q^{n+1} - 2(n+1)e\big) - (1 - q^n - 2ne) = pq^n - 2e.$$

The gain becomes negative at the optimal search depth, giving

$$\boxed{n^* = \left\lceil \frac{\log(2e/p)}{\log q} \right\rceil .} \qquad (p > 2e) \tag{7}$$

The optimal search depth n^* is illustrated in Fig. 3(a.i). In the upper left "triangle", where $p < 2e$, we have $n^* = 0$, i.e., the value of the searched information is too low to justify a search. Note also that when $p \to 1$, i.e., when the content becomes highly available, the optimal search depth is $n^* = 1$ *for any fixed transmission cost* $e < p/2$. This is due to the fact that the content is always found at the first node, and still continuing the search further would just waste energy and time. Indeed, this inability to dynamically stop the search is the Achilles heel of all *static* search strategies.

Case $0 < \gamma < 1$.

Let us next consider unreliable return paths. The condition remains the same, i.e., at the optimal depth n we have $U_{n+1} - U_n \leq 0$. Unfortunately, in this case we cannot express n^* in closed form. However, we can determine the *critical transmission cost* e_n^*,

$$e_n^* = \frac{\gamma^n(q^n + \gamma - \gamma q^{n+1} - 1)}{1 + \gamma^n}$$

which is the smallest transmission cost for which the optimal search depth is $n^* = n$. Conversely,

$$n^* = \arg\min_n \{n \mid e_n^* < e\}.$$

Fig. 3(a.ii) depicts the optimal static search depth when the return path is unreliable and each link backward exists with the probability of $\gamma = 0.7$.

4.1.2 Dynamic strategy

Let us next consider search strategies that adjust the search depth dynamically as the search progresses. In the Bernoulli case, the obvious dynamic search strategy searches at most n nodes (the max. depth) and terminates immediately if the content is found. Hence, e.g., with the probability of p, the first node has the content and the total number of transmissions is 2 (out of which, the latter is successful with probability of γ).

Note that the expected value of the content *found* (but not necessarily successfully returned) is the same as with the static strategy, $\mathrm{E}[\max_i V_i] = 1 - q^n$. However, the search may terminate earlier, which (i) saves in the number of transmissions and also (ii) improves the probability of successfully returning a response.

Case $\gamma = 1$.

Let us again start with the ideal case with $\gamma = 1$. Then the mean number of transmissions is

$$N_n = (2p + 4qp + \ldots + 2nq^{n-1}p) + nq^n = \frac{2 - q^n(2 + np)}{p}.$$

The expected response reaching the source node is given by (3). Hence, the mean utility is

$$\begin{aligned} U_n &:= 1 - q^n - \frac{2 - q^n(2 + np)}{p} \cdot e \\ &= \frac{p - 2e - (p - e(2 + np))q^n}{p}. \end{aligned}$$

The gain from including one more hop, $U_{n+1} - U_n$, becomes negative at the optimal maximum depth n^*. Solving this gives the exact solution for the optimal maximum search depth n^*,

$$\boxed{n^* = \left\lceil \frac{1}{e} - \frac{1}{p} \right\rceil - 1.} \tag{8}$$

Solving for e gives the critical transmission cost,

$$e_n^* = \frac{p}{(n+1)p + 1}.$$

The optimal search depth with the dynamic strategy is illustrated in Fig. 3(b.i). Note that the maximum search depth (but not the mean) with the dynamic strategy is always greater than or equal to the search depth with the static strategy. Moreover, the equicontour lines are strictly increasing functions of the availability p due to the fact that this strategy is able to stop the search dynamically. In passing we note that if we are required to return also a null answer with the Bernoulli case, then the optimal search depth becomes ∞ if $p > 2e$, and otherwise it is zero.

Case $0 < \gamma < 1$.

The forward path remains identical and we can condition on the number of hops travelled before the searched content is found. The probability that the content is found at the ith hop is $q^{i-1}p$, and thus the mean value of the response is

$$R_n = \sum_{i=1}^{n} q^{i-1} p \cdot \gamma^i = \frac{(1-q)(1-(q\gamma)^n)\gamma}{1 - q\gamma}.$$

For the cost we need to determine the mean number of hops. To this end, we condition also on the number of transmissions on the return path (out of which the last one may have failed). Let i denote the number of hops the query travels, i.e., the length of the forward path, and j the number of transmissions on the return path. Then the mean number of transmissions is given by

$$N_n = \sum_{i=1}^{n} q^{i-1} p \cdot \left(i + \sum_{j=1}^{i} j\gamma^{j-1}(1-\gamma) + i\gamma^i \right) + nq^n,$$

where the last term corresponds to a search that did not find the content. Subtracting $e\,N_n$ from R_n gives the mean utility U_n. Subsequently, for the difference $\Delta U(n) = U_{n+1} - U_n$ we obtain

$$\Delta U(n) = \left(p\gamma^{1+n} - \left(1 + \frac{p\left(1 - \gamma^{n+1}\right)}{1 - \gamma} \right) e \right) q^n.$$

Similarly as before, determining the root of $\Delta U(n)$ gives the optimal search depth. After some manipulation, we get

$$\boxed{n^* = \left\lceil \log\left(\frac{e(1 - \gamma + p)}{p(1 - \gamma + e)} \right) / \log\gamma \right\rceil - 1.}$$

Alternatively, solving for e gives the critical transmission cost,

$$e_n^* = \frac{p(1-\gamma)\gamma^{1+n}}{p\left(1 - \gamma^{1+n}\right) + 1 - \gamma},$$

which was the minimum transmission cost at which $n^* = n$.

Fig. 3(b.ii) illustrates the optimal search depth n^* as a function of the availability p and transmission cost e for fixed $\gamma = 0.7$. The curves appear as scaled down versions of Fig. 3(b.i).

In Fig. 4, we have fixed $e = 0.1$ and $p = 0.5$, and vary $1 - \gamma$ that corresponds to the loss probability on the links of the return path. We notice that n^* decreases to 1 as the loss probability increases, i.e., when the return path becomes uncertain. As the intuition suggests, *under a high mobility, long paths become fragile and should be avoided, and only the neighboring node(s) should be involved in the search.*

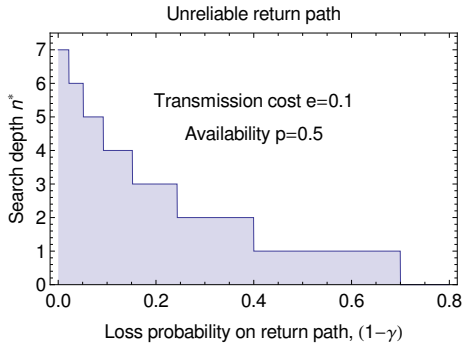

Figure 4: The optimal dynamic search depth with fixed $e = 0.1$ and $p = 0.5$ as a function of $1-\gamma$.

4.1.3 Learning dynamic strategies

All earlier strategies, both static and dynamic, have made the crucial assumption that the value distribution (defining the value of information per node for the query) is known *a priori*. In practice this may not be the case, even though one can envision that empirical distribution have been obtained based on past encounters. In this section, we take the Bayesian approach and refine our estimate of the value distribution as the search progresses further. For simplicity of notation, we assume Bernoulli case with unknown information availability p, but note that the approach itself can be generalized to other value distributions.

A priori we assume that p is uniformly distributed on $(0, 1)$. As the source node does not have an answer, we consider that initially one node has been checked and found out not to have a valid response. Suppose that in state i we have checked i nodes and none of them had a valid response. The Bayes formula gives the conditional pdf for p,

$$f(p \mid i) = \frac{\mathrm{P}\{i \mid p\} \cdot f(p)}{\int_0^1 \mathrm{P}\{i \mid p\} \cdot f(p)\, dp},$$

where $\mathrm{P}\{i \mid p\} = q^i$ and[1] $f(p) = 1$, giving

$$f(p \mid i) = \frac{q^i}{\int_0^1 q^i\, dp} = (i+1)q^i.$$

Subsequently, the expected value of $p = 1-q$ after i negative observations is

$$\hat{p}_i = \frac{1}{i+2}.$$

Case $\gamma = 1$.

Let w_i denote the expected final outcome from state i with optimal strategy, i.e., we have already checked $i+1$ nodes (including oneself) without luck. Then the search strategy bases its action on the assumption that $\hat{p}_i$ is the probability that the ith node has the searched content (i.e., on the condition that none of the previous nodes had it). Recursively this gives

$$w_i = \max\left\{-ie,\ \frac{1}{i+3}(1 - 2e(i+1)) + \frac{i+2}{i+3}w_{i+1}\right\}, \quad (9)$$

where the first case corresponds to terminating the search after i transmissions, and the second case corresponds to continuing the search to the $(i+1)$th node. This otherwise infinite recursion can be constrained by noting that there is no use to continue if the possible gain is less than what it takes to return an answer back to the source, i.e., when

$$(i+2)e \geq 1 \quad \Rightarrow \quad i \geq \frac{1}{e} - 2.$$

That is, for states $i \geq 1/e - 2$, we have $w_i = -ie$. The learning search strategy, based on the Bayesian thinking, thus continues the search as long as the second option in (9) is greater than $-ie$.

In particular, for the maximum search depth n^* we have $w_{n^*} > w_{n^*+1} = -(n+1)e$. Consequently, letting $\Delta U(n)$ denote the gain from continueing the search exactly one step further,

$$\begin{aligned}\Delta U(n) &:= \frac{1}{n+3}(1 - 2e(n+1)) + \frac{n+2}{n+3}(-(n+1)e) + ne \\ &= \frac{1 - 2e(2+e)}{n+3},\end{aligned}$$

we need to find n for which $\Delta U(n)$ becomes negative. Therefore,

$$\boxed{n^* = \left\lceil \frac{1}{2e} \right\rceil - 2.} \qquad (e < 0.25) \quad (10)$$

Comparing (10) to (8), we note that both behave according to $\propto e^{-1}$. The knowledge of the availability of the content, p, affects the factor of e^{-1} term and the constant term. The optimal search depth with the learning strategy is illustrated in Fig. 3(c.i). Note the independence to the availability p, which this strategy does not know.

Case $0 < \gamma < 1$.

Similarly as with the two earlier cases, we can consider unreliable return path also in this case. In particular, we assume that parameter γ is stationary and has been determined when the search is triggered (γ depends on mobility, not on the content searched). Due to space constraints, here we simply give the results. The expected gain in utility from depth n to $n+1$ is

$$\Delta U(n) = \frac{\gamma^{1+n} - 2e(2+n)}{3+n}.$$

In this case, n^*, corresponding to the root of $\Delta U(n)$, cannot be expressed in closed form, but one needs to find $k = 2+n$ that satisfies (cf. Lambert W function)

$$\frac{\gamma^k}{k} = 2\gamma e.$$

However, for the critical transmission cost we have explicitly

$$e_n^* = \frac{\gamma^{1+n}}{2(2+n)},$$

which holds also for $\gamma = 1$. We note that as γ decreases, the critical transmission costs decrease by factor of γ^{n+1} for each n.

Fig. 3(c.ii) illustrates the optimal search depth n^* as a function of the transmission cost e for fixed $\gamma = 0.7$.

4.2 Partial information

Next we assume that some nodes may be able to provide partial answers to a query, i.e., responses that are good but not complete. For example, recent but not current information about football results could be considered as good but not complete answer to a query. For simplicity, in this section we assume ideal return paths with $\gamma = 1$. As example cases, we assume that value of the response from a node obeys either uniform or exponential distribution, for which we derive the optimal static strategies.

4.2.1 Uniform distribution

Suppose first that $V_i \sim \mathrm{U}(0, v_{\max})$, i.e., nodes may have partial answers to the query measured by the *value*. Value $v_{\max}$ corresponds to a complete answer. The CDF of the maximum value among n samples is

$$\mathrm{P}\{\max_i V_i < x\} = \mathrm{P}\{V < x\}^n = (x/v_{\max})^n.$$

[1]Note that the approach allows an arbitrary *a priori* distribution for the availability, which can be in practice based on, e.g., earlier similar queries.

Subsequently, the expected value of the response is

$$\mathrm{E}[\max_i V_i] = \frac{n}{n+1} v_{\max},$$

and the utility reduces to

$$U_n = \frac{n}{n+1} v_{\max} - 2ne. \tag{11}$$

Similarly as in the previous case, one can determine the optimal static search depth n^*. Let q denote the ratio of the maximum value of the response to unit transmission cost, $\beta = v_{\max}/e$. Then it follows that

$$\boxed{n^* = \left\lceil \frac{\sqrt{1+2\beta}-3}{2} \right\rceil.} \qquad (\beta > 4).$$

4.2.2 Exponential distribution

Next we assume that $V_i \sim \mathrm{Exp}(\lambda)$. In this case, the expected value of the response from an arbitrary node is $\mathrm{E}[V_i] = 1/\lambda$, and CDF of the maximum value is

$$\mathrm{P}\{\max_i V_i < x\} = (1 - e^{-\lambda x})^n.$$

It follows that the expected value of the query is

$$\mathrm{E}[\max_i V_i] = \frac{H(n)}{\lambda},$$

where $H(n)$ denotes the nth harmonic number,

$$H(n) = 1/1 + 1/2 + \ldots + 1/n.$$

Our objective is to maximize the expected utility,

$$U_n = \frac{H(n)}{\lambda} - 2ne, \tag{12}$$

Considering again the difference $U_{n+1} - U_n = \frac{1}{\lambda(n+1)} - 2e$ yields the optimal search depth,

$$\boxed{n^* = \left\lceil \frac{1}{2\lambda e} \right\rceil - 1.} \qquad (1/\lambda > 2e) \tag{13}$$

4.2.3 General case

In the previous section, we considered *static* strategies when the value of the content had a continuous distribution. In such a case, a search will never find the complete answer, but has to settle with something that is hopefully sufficiently high. The static strategy suits well to such scenario. Next we will assume a finite set of values and determine the optimal *dynamic search strategy* using dynamic programming. Thus, the decisions may depend also on what has been found so far. Moreover, we allow multiple responses which make more sense in this case, where even a better response can be found later.

We let $z = (m, n, d, b)$ denote **the state** of the search when the query reaches node n, where

- m is the number of transmissions so far
- n is the distance to the source (in hops)
- d is the highest valued response already sent towards the source (by an earlier node)
- b is the highest valued response that node n could send, $b = \max\{V_1, \ldots, V_n\}$

We can write at state $z = (m, n, d, b)$ the (expected) final utility for each action (see the model), and choose the best among them,

$$w(m, n, d, b) = \max\{a_1, a_2, a_3, a_4\},$$

where a_j denotes the (expected) final utility with action j,

$$\begin{cases} a_1 = d - m \cdot e, \\ a_2 = b - (m+n) \cdot e, \\ a_3 = \mathrm{E}[w(m+1, n+1, d, \max\{b, V_{n+1}\})], \\ a_4 = \mathrm{E}[w(m+n+1, n+1, b, \max\{b, V_{n+1}\})], \end{cases}$$

where

$$\begin{cases} a_1 \rightarrow & \text{stop the search,} \\ a_2 \rightarrow & \text{stop the search and send a response back,} \\ a_3 \rightarrow & \text{continue the search further,} \\ a_4 \rightarrow & \text{continue the search, but also send a response back.} \end{cases}$$

The optimal action in state z is given by $\arg\max_j \{a_j\}$. Clearly the actions 2 and 4 make no sense when $d = b$, i.e., when no better response than already delivered is available. The evaluation of the above equations directly leads to an infinite recursion as both a_3 and a_4 are defined in terms of $w(z)$. However, we can exclude both actions when n and m become too large by a simple observation. Namely, one should not forward a query if even the maximum value of the response, denoted by $v_{\max}$, from the next node is not worth the trouble of forwarding and sending back the response, i.e., if

$$\begin{aligned} v_{\max} - (m+n+2)e \le \max\{a_1, a_2\}, &\qquad \text{for } a_3, \text{ and} \\ v_{\max} - (m+2n+2)e \le \max\{a_1, a_2\}, &\qquad \text{for } a_4. \end{aligned}$$

With these, the recursion becomes finite and the optimal actions can be determined for any state. Unfortunately, the number of states still explodes when $e \to 0$, which narrows the usability of the dynamic programming approach at this limit. On the other hand, when $e \to 0$ and V_i obey a discrete distribution, the optimal strategy can neglect the transmission costs.

5. PERFORMANCE EVALUATION

We now evaluate the performance of the developed search strategies with several numerical examples. We start with the Bernoulli case, where a node either has the complete information (e.g. a particular file) to a query or nothing, and then continue with the more complicated scenarios where also partial information exists.

5.1 Bernoulli: Search of a specific content

In our first example, we assume that a node either has the (complete) answer to a query, or nothing, i.e., the Bernoulli case. We compare the learning strategy (see Section 4.1.3) that determines the content availability p during the search to the optimal dynamic strategy that (miraculously) already knows the correct value of p. Our numerical results show that the difference in the performance is typically minimal. In other words, the learning search strategy works very well across many values of p and one only has to know the transmission cost e.

Fig. 5 (left) illustrates the maximum search depth n^* with the learning strategy (bold green line) and the optimal dynamic strategies that know the probability p of a node having the searched content, when $p = \{10\%, 20\%, 50\%\}$. Note

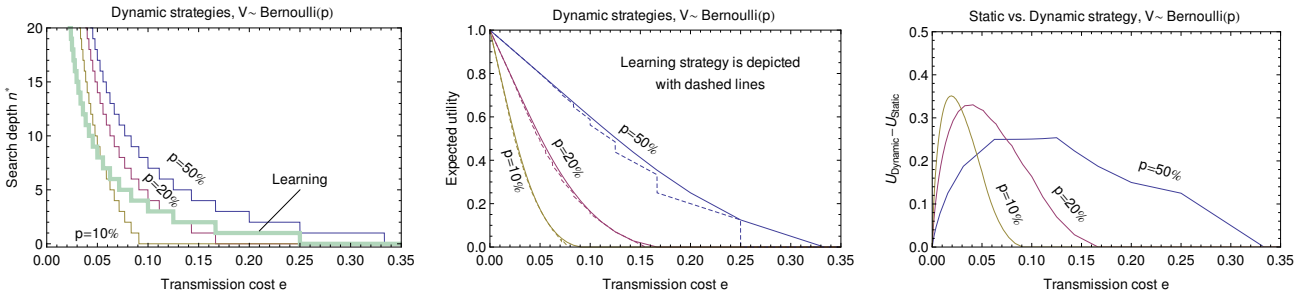

Figure 5: Left: The learning strategy unaware of p vs. dynamic strategies that know p exactly. Middle: Mean utility with the learning and dynamic strategies. Right: Decrease in utility if static strategy instead of the dynamic. ($\gamma = 1$)

that the maximum search depth with the learning strategy is independent of p (by design) and essentially depends only on the ratio of the value of the searched information (normalized to one here) to the unit transmission cost e. We see that the learning strategy is an educated compromise, as expected.

Fig. 5 (middle) illustrates the resulting performance, i.e., the mean utility of a search as a function of the transmission cost e in the three cases, $p = \{10\%, 20\%, 50\%\}$. The dashed lines correspond to the performance with the learning strategy, and the solid lines to strategy that is aware of the correct value of p. We notice that the difference is negligible as soon as $p < 50\%$, or $e < 0.25$. In particular, when p is smaller than 50%, a typical search involves several nodes before the content is found, and during this time a good estimate for p becomes available, which explains the observed good performance of the learning strategy.

Static policy, included merely for comparison, cannot be expected to perform well. Fig. 5 (right) shows the difference in the expected utility when compared to the dynamic strategy. We note that the difference is considerable when the absolute values vary from zero to one.

Next we consider the performance penalty in terms of the mean utility U_n due to an unreliable return path characterized by the parameter γ. Fig. 6 illustrates the equivalue contours of the utility in the ideal case with $\gamma = 1$ (solid lines) against the setting where $\gamma = 0.7$ (dashes lines). We can observe that the performance deteriorates when links become unreliable (e.g., due to mobility), but, at least with $\gamma = 0.7$, the performance loss is reasonable given the search algorithm takes γ into account. This suggests that a well executed search makes sense also in low to moderate mobility scenarios corresponding to $\gamma = 0.7$.

5.2 Partial information: Continuous case

Let us then consider the case where nodes can provide good but incomplete answers to a query. We study the static search strategies where the search depth is fixed to n. That is, the query is forwarded to the distance of n, and then the best response found is returned along the same path back to the source (with $\gamma = 1$). The utility of such a search was $\max\{V_1, \ldots, V_n\} - 2ne$. We consider the following two value distributions discussed already in Section 4.2:

- Uniformly distributed values, $V_i \sim \mathrm{U}(0, 1)$
- Exponentially distributed values, $V_i \sim \mathrm{Exp}(2)$

Note that $\mathrm{E}[V_i] = 0.5$ in both cases.

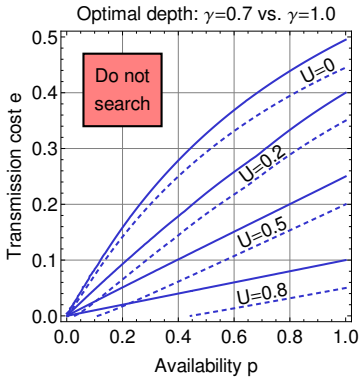

Figure 6: Equivalue curves for the utility $\mathbf{U_{n^*}}$ with the dynamic strategy when $\gamma = 1$ (solid lines) and $\gamma = 0.7$ (dashed lines).

Fig. 7 (left) depicts the optimal search depth n^* as the function of the unit transmission cost e. We can see that as $e \to 0$, the optimal search depth goes again to infinity, as expected. With exponential distribution, the value of the response is not limited and the optimal search depth n^* is somewhat higher than with the uniform distribution. We also note that as $e > 1/4$, i.e., when $2e > \mathrm{E}[V_1]$, the cost of a transmission is too expensive and one should search only the "own pockets", $n^* = 0$.

Fig. 7 (right) depicts the mean utility with the optimal search strategies. We can see that with the finite valued distribution, $V_i \sim \mathrm{U}(0, 1)$, the utility converges to 1 as the transmission cost goes to zero, $e \to 0$. In contrast, with $V_i \sim \mathrm{Exp}(2)$ the utility is unbounded in this limit. In general, the shapes are similar to the Bernoulli cases depicted in Fig. 5.

5.3 Dynamic strategy with partial information

Finally, we assume that a response to a query may have four different values: no information (0), related (1), good response (2) and a perfect answer (3). In particular, we assume discrete values $V_i \sim \mathrm{U}(0, 3)$, i.e., V_i obtains values $0, 1, 2, 3$ uniformly in random. As explained in Section 4.1.2, in this case we can determine the optimal dynamic search strategy (*Dynamic*). This is a sophisticated strategy that may return also with partial information. As reference strategies, we consider also the *static* strategy that searches always n nodes and returns the maximum response

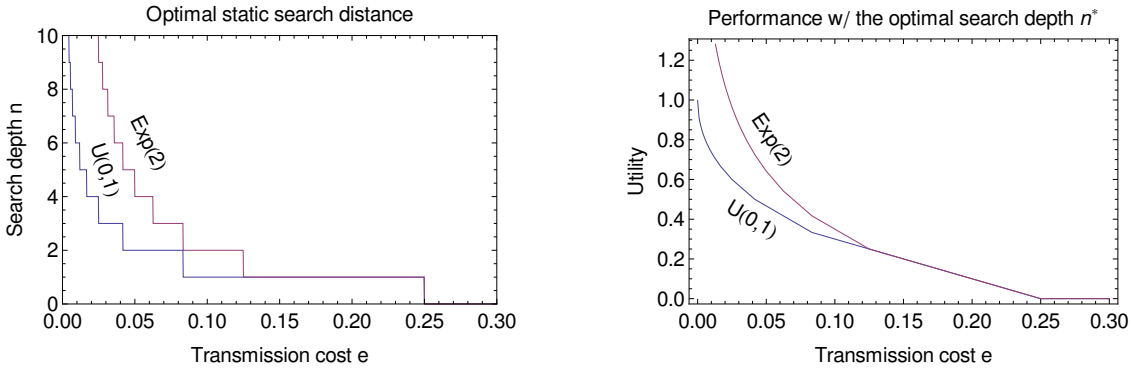

Figure 7: Optimal search depth n^* with static strategies for two continuous value distributions.

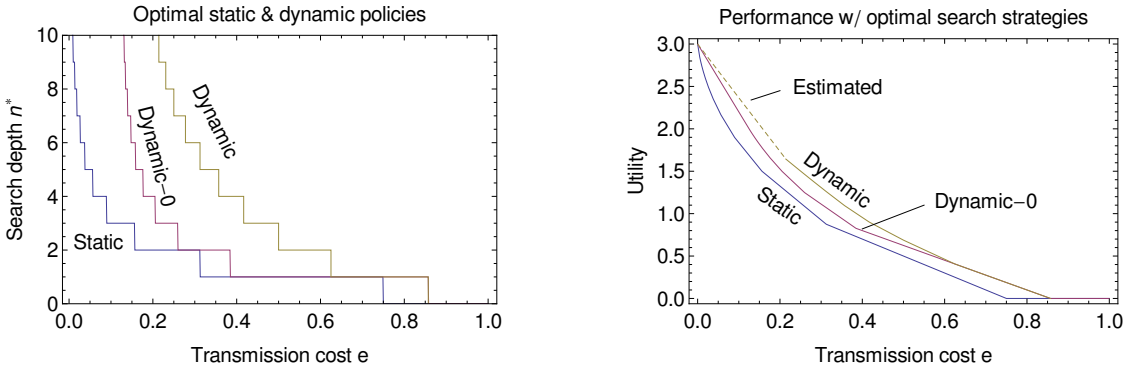

Figure 8: Optimal dynamic vs. static strategy with a discrete $V_i \sim U(0, 3)$.

found (even if zero value), and the *Dynamic-0* strategy that returns immediately if the perfect answer has been found ($V_i = 3$) and does not send a response at all if nothing useful was found.

Fig. 8(left) illustrates the resulting optimal strategies. On the x-axis is the transmission cost e, and the y-axis corresponds to the maximum search depth n^*. Note that with the optimal dynamic strategy, this depth n^* is sought (at least) if all earlier $n^* - 1$ nodes had no information, i.e., if $V_i = 0$, $i = 1, \ldots, (n-1)$. We can see that the dynamic strategies cover a wider distance than the static one (when needed), as expected. Fig. 8(right) depicts the resulting performance in terms of the mean utility per search. We can observe that the dynamic search makes sense even when e is a bit higher. The absolute gain is about the same for a quite large range of unit transmission costs e. At the limit when $e \to 0$, all schemes will search until the best possible response ($V_i = 3$) is found, i.e., at the limit $e \to 0$ all curves converge to 3. As e increases, the mean performance behaves as a convex function, until at some point $n^* \to 0$.

6. DISCUSSION

In our model, we focused on the actions that a single query takes along the search path. However, in a mobile opportunistic network, rather than a single linear path, the search (with replicated queries) can be represented as a union of several paths. Search along multiple paths may implicate two points: (i) the average distance to valuable response is likely to get shorter, (ii) each decision (i.e., to forward or not, to respond or not) is based on partial information, i.e., on what has happened along its path, whereas the state of the other search path is unknown and can only be estimated. For the former, we plan to explore the average distance, i.e., the hop count, from the searching node that the search should be expanded for the most efficient search. In this work, we have provided the optimal hop count using the theoretical formalism along with several simplifications. In our future work, we would like to focus on a more relaxed scenario, e.g., realistic mobility models, and discover the optimal hop count by the help of simulations. For the latter, each node on the search path can use its observations of the network as in [9] (e.g., the number of nodes that has received the search packet and possibility of a response) to predict the state of the search, e.g., utility.

Note that a static strategy with a multiple concurrent search paths (returning a response at the end of each path) has the same cost as in the case of a single path. Thus, also in this case one needs to solve $\max\{V_1, \ldots, V_{n_1+\ldots+n_k}\} - 2ne$. Obviously, if $\mathrm{P}\{V_i = 0\}$ is non-negligible, declining from returning a response with a zero value improves the performance with multiple search paths more than with a single (longer) search path. The mean response time also becomes shorter due to the parallel operation. In any case, the analysis of multipath search needs further investigations.

Additionally, we plan to study models with unequal transmissions costs; e_{fwd} for query and e_{back} for the response. For longer paths, we need to consider also alternative return paths that may take "shortcuts" as the response knows which nodes *were* the closest to the origin initially.

7. CONCLUSIONS

Forwarding packets in a meaningful manner in opportunistic networks is not an easy task. The basic routing schemes such as the plain flooding and the *spray and wait* algorithm [11] try to solve the one directional problem of sending information from a source to a particular destination. In our case, the setting is significantly more challenging because (i) we do not know who has the information, and (ii) the searched content must be delivered back to the searching node. The important contribution of this paper is the analytical treatment of the "self-guiding" search process in wireless ad-hoc networks, where the query takes actions based on its *a priori* information and the observations made during the search. In many cases, we were able to characterize the optimal search strategy that maximizes the expected utility. Despite of the shortcomings of our simplified models, we believe that the similar principles as studied in this paper can be also applied in practice in a more complete setting. In our future work, we will include to the model more realism by adding the option to replicate the query, where the additional challenge comes from the distributed decision making.

Acknowledgements

This work was supported by the Academy of Finland in the PDP project (grant no. 260014).

8. REFERENCES

[1] Aruna Balasubramanian, Brian Levine, and Arun Venkataramani. DTN routing as a resource allocation problem. *ACM SIGCOMM Computer Communication Review*, 37(4):373–384, 2007.

[2] Suzan Bayhan, Esa Hyytiä, Jussi Kangasharju, and Jörg Ott. Seeker-assisted information search in mobile clouds. In *Proc. of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, MCC '13, 2013.

[3] Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapé. Analysis of country-wide internet outages caused by censorship. In *Proc. of the ACM SIGCOMM conference on Internet measurement*, pages 1–18, 2011.

[4] K. Fall and S. Farrell. DTN: an architectural retrospective. *IEEE Journal on Selected Areas in Communications*, 26(5):828–836, June 2008.

[5] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *Proc. of conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–34, 2003.

[6] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot. Pocket switched networks and the consequences of human mobility in conference environments. In *ACM SIGCOMM Workshop on Delay Tolerant Networking*, 2005.

[7] Pan Hui, Jon Crowcroft, and Eiko Yoneki. Bubble Rap: Social-based forwarding in delay tolerant networks. In *Proc. of ACM MobiHoc '08*, pages 241–250, 2008.

[8] Cong Liu and Jie Wu. An optimal probabilistic forwarding protocol in delay tolerant networks. In *Proc. of ACM MobiHoc '09*, pages 105–114. ACM, 2009.

[9] Mikko Pitkänen, Teemu Karkkainen, Janico Greifenberg, and Jörg Ott. Searching for content in mobile DTNs. In *Proc. of IEEE Pervasive Computing and Communications (PerCom)*, 2009.

[10] Nishanth Sastry, D Manjunath, Karen Sollins, and Jon Crowcroft. Data delivery properties of human contact networks. *IEEE Transactions on Mobile Computing*, 10(6):868–880, 2011.

[11] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. In *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, pages 252–259, 2005.

[12] Ying Zhu, Bin Xu, Xinghua Shi, and Yu Wang. A survey of social-based routing in delay tolerant networks: Positive and negative social effects. *IEEE Communications Surveys Tutorials*, 15(1):387–401, First 2013.