On Search and Content Availability in Opportunistic Networks

Esa Hyytiä*, Suzan Bayhan, Jörg Ott, Jussi Kangasharju

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 consider a model where both the query and response follow the same or similar path. We formulate the problem of optimal search for 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. Moreover, we study different methods to *a priori* learn the availability of the content in the network based on passive observations (e.g., using regression and maximum-likelihood based estimates). Such information is highly valuable when defining the optimal search parameters. Finally, we investigate the principal factors affecting the optimal search strategy.

Keywords: Mobile opportunistic networks, mobile search, mobile cloud computing, dynamic programming, availability estimation

1. Introduction

Mobile users rely on cloud-based third party services for information sharing and messaging even if they are close by so that their respective devices could exchange information directly, without taking a long distance detour across the Internet and potentially half-way around the world. This holds for a broad spectrum of services from email and chat (e.g., Jabber) to online social networks (e.g., Facebook, Twitter) to data and file sharing (e.g., Dropbox). Important features of such services include that they are well-known and well-managed (including backups), support a global community, and are usually instantly accessible, so that users have them always conveniently at their disposal—as long as they are connected to the Internet.

However, this convenience creates dependencies, at the very least on Internet access and the availability and reachability of the respective third-party service. It also creates cost: the data needs to be moved (repeatedly) through parts of the Internet, consuming network and energy resources, and needs to be processed and stored in the cloud—including backups, as even short-lived data is rarely deleted (immediately or at all) after sharing, as the authors observe in their and others use of, e.g., Dropbox.

Instead of using infrastructure services, nodes can directly exchange content via short-range communication interfaces, e.g., Wi-Fi, Bluetooth, such that only the peers in wireless contact are involved. In this manner, peers can build a network operating in an ad hoc mode and facilitating the communication between non-adjacent nodes via hop-by-hop data forwarding. With the sheer growth of data traffic and fierce competition for the bandwidth, operators benefit from this approach for decreasing the path between the content provider and the consumer. On the other hand, due to the instability of the direct links, this scheme may not guarantee certain delay bounds, limiting its applicability to

☆An earlier version of this paper was presented at ACM MSWiM 2014 [13].

*Corresponding author, email: esa.hyytia@aalto.fi

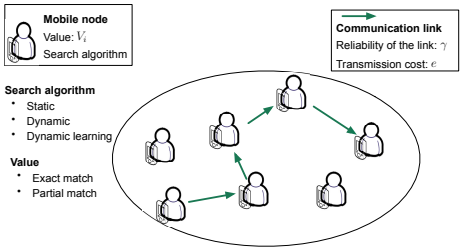


Figure 1: Search query travels from a node to another and the path forms a linear trajectory in space. The response is assumed to follow the same (or similar) path backwards.

only delay-tolerant applications. *Delay-tolerant networking* (DTN) [8, 9] 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.

Mobile opportunistic networks, also dubbed *Pocket switched networks* [19][11], are of particular interest with the increasing diffusion of powerful mobile wireless devices (such as tablets and smartphones). Mobile devices carried by humans can exchange information 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 [7].

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 (or similar) path backwards. That is, the response path is assumed to equally long, and moreover, it can 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.

The main contributions of this paper are as follows:

1. We provide theoretical modelling of several search strategies: static, dynamic, and learning dynamic search. In *static search*, the search is extended to a predefined number of nodes n whereas in *dynamic strategies* a node 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.
2. To assist the search, we also study different methods to *a priori* estimate the content availability based on other similar queries nodes have observed in the past. As these methods rely on passive observations, the only additional cost is a negligible increase in the computational effort, while the benefits can be considerable when each query has a good understanding of the operating environment from the beginning. To the best of our knowledge, our paper is the first paper on content availability estimation in an opportunistic network.
3. For both the static and dynamic search, we model the search process under both *exact matching* and *partial matching* content items. The former corresponds to search of the specific content (yes or no), whereas in the latter a multitude of answers is possible each with a different value.
4. Although we assume that response follows the same or equally long 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 (cf. mobility).

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. Section 4 presents the analysis of different search strategies, whereas Section 5

discusses four methods to *a priori* estimate the average content availability in the opportunistic network. These are followed by a performance evaluation 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 sought 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) [3] vaguely maps a content to the nodes of a particular community which is a group of nodes sharing common interests. Hui et al. [11], 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 [12], which tries to exploit the social structures when making the forwarding decisions.

Rational search schemes should direct the search towards the nodes that have a higher likelihood of having the sought item. On the way to these "potential content providers", nodes with good relaying capabilities can be employed as intermediate carriers. The forwarding decision can exploit various characteristics of the network, e.g., centrality of the nodes, (sub-)groups in the network, content and node relevance. For example, SAS [3] exploits the *homophily principle*, tendency to associate and interact with similar others, and directs the search towards the nodes of the same community as the content might have been sought and be readily available at a node in this community. In order to avoid searching only a specific part of the network, SAS expands the search also to elsewhere, although the probability of finding the content might be lower. Likewise, DelQue [10] defines *geo-community* concept to associate the interests with the locations (e.g., people interested in basketball contact each other in gyms). 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 nodes issue queries for finding specific contents.

Deciding when to stop the forwarding of the search query is another challenge as nodes operate in distributed fashion relying on their local knowledge. Although a query might have reached the content provider and a response message may be already on the way, this may not be signalled immediately to 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. [18] define a termination logic in which each node using the observed degree of itself estimates the number of nodes the query might have reached by now and the number of possible responses generated by these nodes. The query is terminated if the estimate is above some threshold. [23] leverages the self-declared expertise of a node for each query category (e.g., history or arts) to decide whether this node's response is sufficiently precise for the received query. If node's expertise – defined as the probability of correctly answering a query – is above a threshold, the node prepares a response and the query is deleted with some probability. The probability depends on node's expertise and lets some redundancy to account for the inaccuracy in self-declared expertise. 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. [23] calculates the "k-hop reached expertise" to calculate the utility of each node for multi-hop search and experimentally assesses the performance of k-hop search for $k \leqslant 3$. DelQue in [10] limits the search to two hops based on the observation that nodes with their one-hop neighbors can cover a significant portion of the network. As for search termination, 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 bidirectional 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

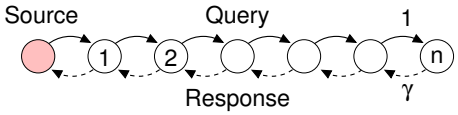


Figure 2: Linear network, where search query travels to the right and a possible response(s) to the left.

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 explicitly. SAS considers a direct-delivery scheme for the response path whereas DelQue limits the number of relays to one and the relay is selected based on its capability to both deliver the query and also the response back to the searching node. In [23], basic focus is on the query path and they borrow one of the one-to-one routing schemes in the literature to implement the return path. Designing a short response path eases the search process to some extent but may also be limiting the search. Hence, we do not expose any limitations on the path length. The only assumption in our paper is that response follows the same path as the query. 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 and basic terms we use throughout the paper are defined 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.
- Forward path: A query travels on the *forward path*, where the loss probability is assumed to be zero. In other words, a node meets another node within a reasonable time with a high probability.
- Return path: A possible response travels in the opposite direction on the *return 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 there are many reasons for $\gamma < 1$. For example, a node may have carried the search query away, or a previous hop may have gotten out of transmission range.
- *Value of a node* (V_i): For each query, each node i has a response whose 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.
- Let the total number of transmissions be m when the search has completed, and d be the highest valued response that is returned back to the source. Independent of the status of transmission, i.e., a failed transmission attempt or successful one, every transmission attempt on the return path is included in m.
- Each transmission costs e (say energy and time), which is assumed to be the same for both the query and response for simplicity. In practice, queries are expected to be smaller message units than the responses as they are simple text messages. Responses might have a higher cost if they return a lot of data (e.g., music, photos, or video).
- *Critical transmission cost* is the smallest transmission cost for which the optimal search depth is n.
- 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 resulting in $n = m$, and the transmission costs are ne.

- As the search success criterion, we consider the net profit of a search, i.e., *the utility*. We calculate the utility of a search as the value of the response minus the expenses,

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

  In other words, the aim is to maximize the net profit for each search *individually*, and thereby maximize also the global profit rate in the network (cf. social optimality).

- We define the following four actions for a node i along the search path:
  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., [16] and [24]. 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. First, we consider static strategies, where the searching node sets at the time of search admission how many hops the query should go further. In other words, the actions of other nodes are already decided by the searching node, i.e., search and forward the query till the hop limit is reached. Next, we introduce dynamic strategies, where the action of each node 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. In the following, we model these search schemes for two settings. First, we consider a setting in which a node may have the sought item or not. We call this as *exact-match* scenario. Next, we model the content items that may partially match the search query, i.e., the content may not be the perfect answer for the search query but provides some relevant information. We call this case as *partially-match* scenario.

4.1. Exact-Matching Contents: 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 sought content. We refer to p as the *content availability*, and q denotes the probability of the opposite case, $q = 1 - p$. To account for the effect of mobility on the stability of the links, we let the links on the return path be unreliable ($\gamma < 1$). In our model, at most one response is sent per query.

4.1.1. Static strategy

Let us assume that nodes are aware of the content availability p. 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. Given that each link is up and ready for transmission with probability γ independent of other links, we calculate the number of transmissions on the return path as:

$$r_n = \sum_{i=1}^{n} i\gamma^{i-1}(1-\gamma) + n\gamma^n. \tag{2}$$

Based on γ, we define r_n as:

$$r_n = \begin{cases} \dfrac{1-\gamma^n}{1-\gamma}, & \text{when } 0 < \gamma < 1, \\ n, & \text{when } \gamma = 1. \end{cases} \tag{3}$$

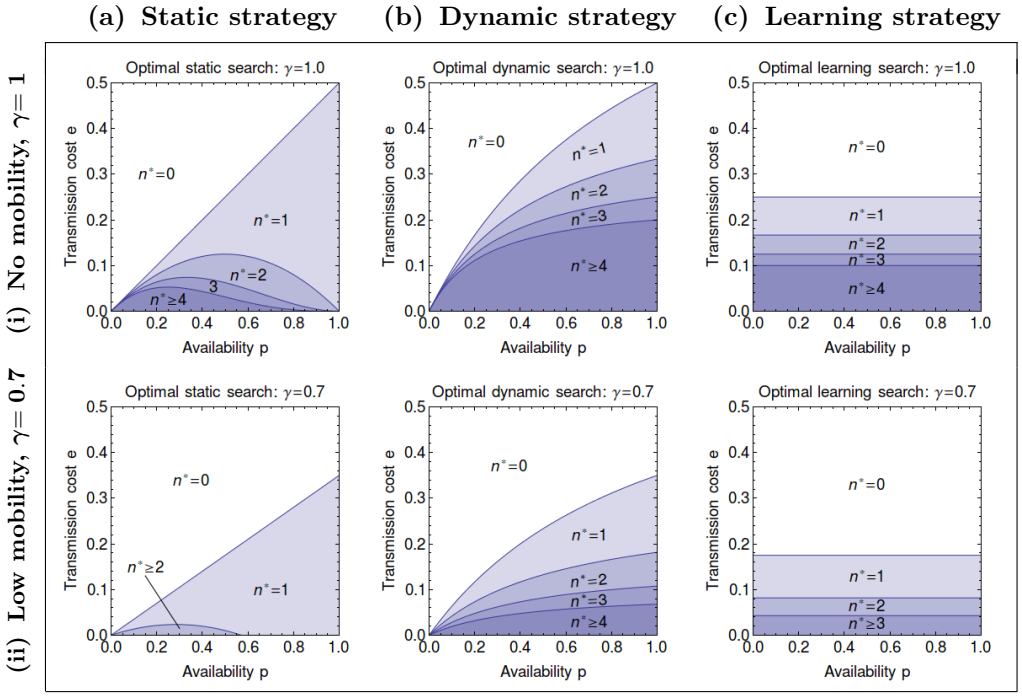


Figure 3: Optimal max. search depth n^* in Bernoulli case ($v_{\max} = 1$) with (a) the static strategy, (b) dynamic strategy and (c) learning strategy (which does not know p *a priori*). The top row corresponds to the ideal case with $\gamma = 1$, whereas on the bottom row the return path is unreliable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

$$\begin{aligned} R_n &= \mathrm{E}[\max\{V_1, .., V_n\}] \cdot \gamma^n \\ &= (0 \cdot q^n + 1 \cdot (1 - q^n))\, \gamma^n \\ &= (1 - q^n)\gamma^n. \end{aligned} \tag{4}$$

Thus, the expected utility under n hop search is

$$U_n = R_n - (n + r_n)e = (1 - q^n)\gamma^n - (n + r_n)e. \tag{5}$$

For $\gamma = 1$, the response travels the same path as the query and hence $r_n = n$. This case also provides the upper bound of the search success for this strategy:

$$U_n = 1 - q^n - 2ne. \tag{6}$$

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

$$n^* = \arg\max_{n \in \mathbb{N}} \quad U_n. \tag{7}$$

We note that this is clearly a non-optimal strategy: if Node 1 already has the sought 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. Below, we provide the optimal hop count for both the perfectly reliable response links (i.e.,$\gamma = 1$) and lossy response links (i.e.,$\gamma < 1$).

- 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

$$\boxed{n^* = \left\lceil \frac{\log(2e/p)}{\log q} \right\rceil}. \qquad (p > 2e) \tag{8}$$

- 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,

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

The optimal search depth n^* for $\gamma = 1$ 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 sought 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.

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 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.

Response value depends on both the value distribution and the successful delivery of this best response. The former is the same as the ideal case whereas the latter depends on γ as well as on which hop the content provider is discovered. Therefore, we need to condition on the hop distance of the node that provides the content. Dynamic strategy enables immediate termination of the query upon discovery of the content. Therefore, if the ith hop is the content provider, it implies that all previously visited $(i-1)$ nodes do not have the content. In other words, 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

$$R_n = \sum_{i=1}^{n} q^{i-1}p \cdot \gamma^i = \frac{(1-q)(1-(q\gamma)^n)\gamma}{1-q\gamma}. \qquad (0 < \gamma \leq 1)$$

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

$$N_n = \sum_{i=1}^{n} q^{i-1}p \cdot \left(i + \sum_{j=1}^{i} j\gamma^{j-1}(1-\gamma) + i\gamma^i \right) + nq^n, \qquad (0 < \gamma \leq 1)$$

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

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

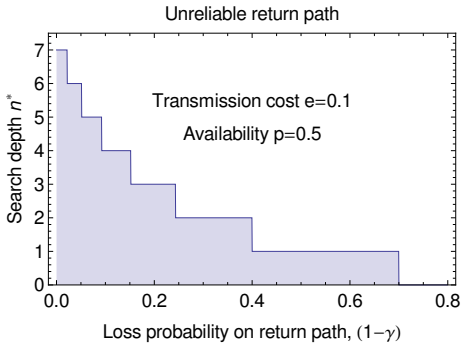


Figure 4: The optimal dynamic search depth with fixed $e = 0.1$ and $p = 0.5$ as a function of $1 - \gamma$.

Determining the root of $\Delta U(n)$ gives the optimal search depth.

- Case $\gamma = 1$: In this case, we obtain explicitly

$$\boxed{n^* = \left\lceil \frac{1}{e} - \frac{1}{p} \right\rceil - 1,} \tag{9}$$

  whereas the corresponding critical transmission cost is

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

- Case $0 < \gamma < 1$: Similarly, for $0 < \gamma < 1$, the optimal search depth is

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

  and the corresponding critical transmission cost is

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

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

The optimal search depth with the dynamic strategy and $\gamma = 1$ 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. 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.*

4.1.3. Learning dynamic strategies

All earlier strategies, both static and dynamic, make 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 has been obtained from 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

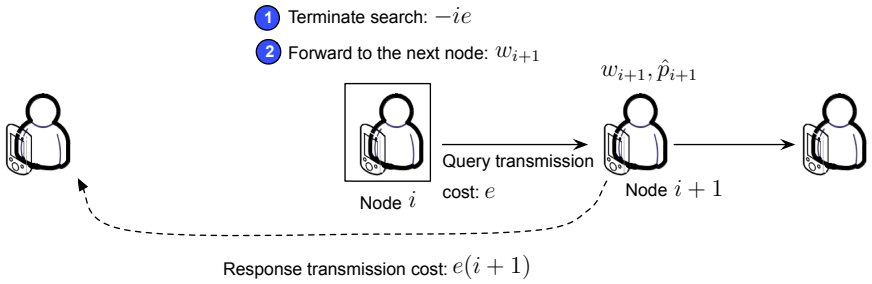


Figure 5: Node i decides on whether to terminate the search or to forward it to the next node depending on its expected outcome and belief of $\hat{p}_{i+1}$ – the probability of finding the content in the next node.

of notation, we assume Bernoulli case with unknown content 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 sought content (i.e., on the condition that none of the previous nodes had it). There are two possible actions at this point– either terminate the search or spread it further. For the first case, there is no reward as the search could not find the content. The corresponding outcome is $-ie$ (see Fig. 5). For the second case, the expected outcome of continuing the search to the $(i+1)$th node is:

$$\hat{p}_{i+1}(1 - 2e(i+1)) + (1 - \hat{p}_{i+1})w_{i+1} = \frac{1}{i+3}(1 - 2e(i+1)) + \frac{i+2}{i+3}w_{i+1},$$

  where the first term denotes the reward of successful discovery of the content; and the second term corresponds to the case that the content is not found at the $(i+1)$th node and search can be spread to the next node or terminated. Recursively w_i is defined as follows:

$$w_i = \max\left\{-ie,\ \frac{1}{i+3}(1 - 2e(i+1)) + \frac{i+2}{i+3}w_{i+1}\right\}. \tag{10}$$

  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. In case the content is discovered at the $(i+1)$th node, the cost of transmitting the response along this path is $(i+1)e$. The node takes into account the additional

[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.

cost of transmitting the query to the next node which then results in the total cost $(i+1)e + e = (i+2)e$. For the node to have some incentive to proceed the search, this cost has to be smaller than the maximum gain, i.e.,

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

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 (10) 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 continuing the search exactly one step further,

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

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

$$\boxed{n^* = \left\lceil \frac{1}{2e} \right\rceil - 2.} \qquad (e < 0.25) \tag{11}$$

Comparing (11) to (9), we note that both behave according to $\propto e^{-1}$. The knowledge of the content availability p affects the factor of e^{-1} term and the constant term.

- Case $0 < \gamma < 1$: Let us assume that parameter γ is stationary and has been determined when the search is triggered (γ depends on mobility, not on the content sought). For brevity, here we simply give the results. The expected gain in utility from depth n to $n+1$ is

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

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

$$\frac{\gamma^{n+2}}{n+2} = 2\gamma e.$$

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.

The optimal search depth with the learning strategy is illustrated in Fig. 3(c.i) for $\gamma = 1$, and in Fig. 3(c.ii) for $\gamma = 0.7$. Note the independence to the content availability p, which this strategy does not know.

4.2. Partially-Matching Contents

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

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

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

$$U_n = \frac{n}{n+1} v_{\max} - 2ne. \tag{12}$$

Similarly as in the previous case, one can determine the optimal static search depth n^*. Let β 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

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

The expected value of the query is

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

where $H(n)$ is the nth harmonic number, $H(n) = 1/1 + \ldots + 1/n$. The expected utility is

$$U_n = \frac{H(n)}{\lambda} - 2ne. \tag{13}$$

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

$$\boxed{n^* = \left\lceil \frac{1}{2\lambda e} \right\rceil - 1.} \qquad (1/\lambda > 2e) \tag{14}$$

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. The decisions may depend also on what has been found so far. Moreover, we allow multiple responses which make sense in this case, where better responses can be found later.

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

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

At each state, i.e., upon reaching the next node, the possible actions are

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

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,

$$w(m, n, d, b) = \max_i \{u(a_i)\},$$

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

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

The optimal action in state z is given by $\arg\max_a\{u(a)\}$. Clearly the actions a_2 and a_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 $u(a_3)$ and $u(a_4)$ 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

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

With these, the recursion becomes finite and the optimal actions can be determined. Unfortunately, the number of states still explodes when $e \to 0$, which narrows the usability of the dynamic programming approach at this limit. However, at this limit the optimal strategy is trivial: search until the response with the highest value $v_{\max}$ has been found.

5. Passive Estimation of the Content Availability

While routing the messages, each node can inspect the message's header to infer the network state. The query message's header consists of several fields including the searching node id, query definition (e.g., an explicit content id, keywords, etc.), and the hop count the message has travelled so far. Similarly, a response message's header carries information about the query, as well as the node that has generated the response. Passively inspecting the encapsulated information, nodes can estimate the content availability to some extent.

Search schemes can obviously benefit from such *a priori* information. In this section, we discuss different methods to estimate the availability based on past observations. As all schemes we consider exploiting already available information, we call them as *passive* schemes. We discuss the performance and shortcoming of these passive schemes in this section, and leave the more advanced *active* schemes as future work. However, first we need to define what we mean by availability. There is at least three possibilities:

1. **Global availability**, denoted by $p^{(G)}$, is defined as the probability that a random node has a content that fulfils a random search criterion.
2. **Type-specific availability**, denoted by $p^{(T)}$, refers to the fraction of nodes that can respond to a random query of the given type. Here we assume that search queries can be categorized to *types*, which could refer to songs of certain genre, sport news, information about traffic delays, etc.
3. **Content-specific availability**, denoted by p_k, refers to the availability of a specific content c_k. In this case, we assume that there are M unique contents in the network.

If a query is about searching a specific content c_k, and p_k or its estimate is available, then the search strategy should utilize that information. If the content-specific availability is unknown, but one has an estimate for the availability of the given type of content, then the search scheme can resort to that. The global availability $p^{(G)}$ would then be the last resort. In contrast, if a query is about a certain type of content (say a pop song with some keywords), then the first option is to utilize the content type specific availability, and if that is not possible, then the global availability.

In summary, the availability itself is already a complicated concept and in order to keep the discussion concise, we implicitly assume a content-specific availability in the rest of this section. At the same time it is good to keep in mind that most of the estimation methods discussed can be also used to estimate $p^{(G)}$ and $p^{(T)}$ without any changes.

Since nodes may have different observations depending on their position in the network, we will use indices to refer to the individual node estimations, e.g., $\hat{p}_k(n_i)$ for n_i's estimation. Let $\mathbf{p} = [p_k]$ be the vector of content availabilities, and $\hat{\mathbf{p}} = [\hat{p}_k]$ be the vector of estimated content availabilities. We assess the estimation error as the absolute difference between $\hat{\mathbf{p}}$ and $\mathbf{p}$, and calculate the estimation error of n_i as follows:

$$\delta_p(n_i) = \frac{\sum_{k=1}^{M} |\hat{p}_k(n_i) - p_k|}{M}. \tag{15}$$

In an active scheme, nodes can share their estimates, and thereby improve their knowledge about the availabilities. Considering the whole network of N nodes as a single entity, we can also calculate the collaborative estimates as the average of the estimates:

$$\hat{p}_k = \frac{\sum_{i=1}^{N} \hat{p}_k(n_i)}{N}. \tag{16}$$

Similar to (15), we calculate the average estimation error denoted by δ_p.

Next we will discuss how each node n_i can estimate the global, type-specific or content-specific availability. Therefore, for clarity, we drop n_i and k from the notation. The first two schemes extract the information from response messages, whereas the other two schemes focus on query messages.

5.1. Extracting availability by counting providers (R-CP)

Let us start with an estimation scheme based on counting the number of content providers observed. To this end, a node can keep track of the set of unique content providers[2] of c_k denoted by S. With this information, a node can calculate the availability of c_k as follows:

$$\hat{p} = \frac{|S|}{N}. \tag{17}$$

As for the complexity of this approach, a node may store up to $N \times M$ entries, which happens when all nodes hold all contents and they are all discovered by the given node.

Strictly speaking, this method would be practical only when the network is small and it makes sense to talk about the number of nodes N. However, generalization for larger and/or dynamic networks is straightforward when one interprets N as the (estimated) mean size of the "local network" that is relevant to search operations of the given node. Henceforth, we refer to this approach as **R-CP** as it relies on the information provided by responses (R), more particularly content provider (CP) id encapsulated in the responses.

5.2. Extracting availability from hop counts (R-HC)

As we showed in Section 4.1.3, a node (or "query") can estimate p after i nodes have sought in their local storage to no avail by inspecting the number of hops a query has travelled. The first node receiving the query estimates the content availability as $1/(1+2) = 0.33$; the second as $1/(2+2) = 0.25$; and so on. Consequently, this method may (initially) lead to *optimistic* estimation of $\hat{p}$, i.e., $\hat{p} > p$.

Additionally, considering more general network topologies, the nodes can receive multiple queries for the same content from diverse paths. Queries following different paths result in different estimations about the availability. Nodes can exploit the diversity in the message hop counts to have more reliable estimates without any extra transmissions.

Compared to information a query carries, a response message may be more informative: it carries information about the content provider as well as the content availability observed by the content provider. In a *response-hop count based estimation*, the content provider adds a field to the response message that shows the number of hops the query travelled till reaching this content provider. Each node on the response path records the number of response messages it has relayed with a specific hop count. Let R_h denote the number of response messages a node has observed (of the given content) that were generated by a content provider who was h hops away from the searching node. Let us denote the observations of the node by vector $\mathbf{R} = [R_h]$. Using this vector, the node calculates its estimate for the availability as follows:

$$\hat{p} = \frac{\sum_{h=1}^{H} R_h/h}{\sum_{h=1}^{H} R_h}, \tag{18}$$

where H is the maximum number of the hop counts the node observes from the response messages. The space complexity of this approach is $H \times M$ (per node), where M is the number of contents (or content types). We refer to this approach as **R-HC** as it relies on the response messages, more particularly hop counts of the queries associated with the response message.

[2]Unlike other estimation schemes, this scheme does not lend itself to global or type-specific availability.

5.3. Regression based estimation of availability (Q-LS)

In this section, we derive a rather sophisticated method to estimate the availability from query messages. Suppose that a node has recorded all (matching) queries that have passed through it. Unlike in earlier schemes, here we neglect the responses as their frequency depends highly on the availability and also on the γ parameter. Moreover, in practice the return path may not be the same as the forward path, and therefore a node may see only a fraction of responses for the queries that it has forwarded.

What a node then can deduce from the number of queries it has observed? We recall that it does not see a query (i) if it does not belong to the presumed route of the query, or (ii) if any node earlier along the route had the sought content and sent a response back. Consequently, it is less likely for a query to reach a given node further it comes from. Our approach utilizes this property in a novel way to estimate the content availability.

In particular, here we assume that the information available is a sample vector $\mathbf{n} = (n_1, \ldots, n_H)$ from a random vector $(N_1, \ldots, N_H)$, where n_i and N_i denote the number of queries observed that have travelled i hops before reaching a given node. We note that, intuitively, if the availability is high, then n_H should be small when compared to n_1, i.e., $\{n_i\}$ is likely to be a decreasing sequence if integer numbers. Specifically, $\mathrm{E}[N_i] > \mathrm{E}[N_{i+1}]$. The key idea is to utilize the differences in the n_i to estimate the availability.

To this end, we need to make some further assumptions. First, we let random number M_i denote the total number of queries that would have reached a given node after i hops if a matching content did not exist in their paths. We assume that each of the M_i queries reaches the given node, independently of the other queries, with the probability of $(1-p)^{i-1}$, where p is the availability of the given content type. Hence,

$$\mathrm{E}[N_i] = (1-p)^{i-1}\mathrm{E}[M_i],$$

and therefore we have an obvious estimator for $\mathrm{E}[M_i]$,

$$\hat{m}_i = \frac{n_i}{(1-p)^{i-1}}.$$

However, we do not know p and actually want to estimate it based on our observations. To this end, we next assume an uniform source distribution,[3] i.e.,

$$M_i \sim M, \tag{19}$$

so that we can write

$$\underbrace{\log(\mathrm{E}[N_i])}_{=y_i} = \underbrace{(i-1)\log(1-p)}_{=b(i-1)} + \underbrace{\log(\mathrm{E}[M])}_{=c},$$

which gives us a linear model, $y_i \approx b(i-1) + c$. Proceeding with the method of least-squares-fit, we write

$$e = \sum_{i=1}^{H} (y_i - b(i-1) - c)^2,$$

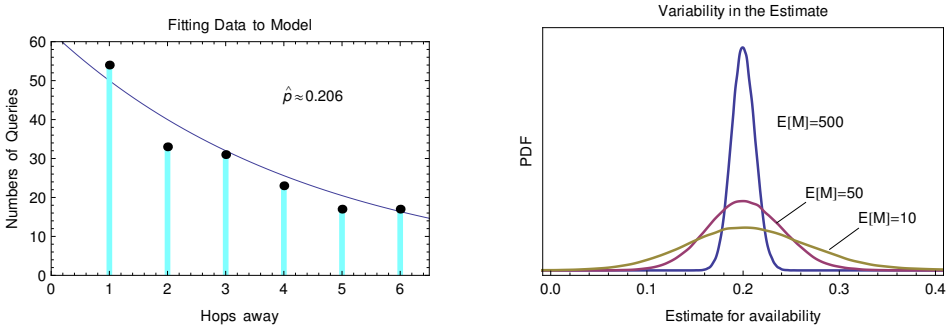
where $y_i = \log n_i$ (assuming $n_i > 0$). Taking the partial derivatives with respect to parameters b and c yields a system of linear equations,

$$\begin{cases} \dfrac{\partial e}{\partial b} = 0, \\ \dfrac{\partial e}{\partial c} = 0. \end{cases}$$

This system can be solved in straightforward fashion, which gives, after some manipulation, an explicit expression for parameter $b = \log(1-p)$,

$$b = \frac{6\sum_{i=1}^{H}(2i - H - 1)y_i}{H(H^2-1)}.$$

[3]This assumption can be relaxed and adjusted if there is some *a priori* information about the topology and the distribution of the M_i. In Section 6.1, we provide some discussion on the distribution of M_i for real world networks.



(a) Regression (b) Variability in the estimate

Figure 6: Regression based approach builds on the vanishing number of queries from n hops away (a), and its quality improves significantly as the number of observed queries increases (b).

Our estimate, based on the past observations $\mathbf{n}$, for the availability p is then

$$\hat{p} = 1 - e^b = 1 - \prod_{i=1}^{H} n_i^{\left(\frac{6(2i-H-1)}{H(H^2-1)}\right)}. \tag{20}$$

We note that in above we have not imposed any constraints on the value of b, and therefore $\hat{p}$ can give infeasible values (less than zero, or greater than one). In such cases, the size of the sample set is either too small, or some of our assumptions, such as (19), simply does not hold. We will return this question later in Section 6. We refer to this approach as **Q-LS** as it relies on the query messages and uses least-squares regression for estimation.

Figure 6(a) illustrates the regression based estimation method. We have obtained a sample vector $\mathbf{n}$ of queries $1, \ldots, 6$ hops away from a given node. The node fits the model parameters to the data using (20) and obtains an estimate for the availability $\hat{p} = 0.206$, when the correct value would have been $p = 0.2$. For this example, we assumed that the $M_i \sim \text{Poisson}(m)$ with $m = 50$.

Figure 6(b) illustrates the variability in the estimate for $m \in \{10, 50, 500\}$. The x-axis corresponds to the value of the estimate $\hat{p}$ and the y-axis is the probability density. We can see that the quality of the estimate is already reasonable when $m = 50$, but as more samples become available ($m = 500$), the error margin becomes negligible (for our purposes).

5.4. Maximum-likelihood estimate for availability (Q-ML)

Let us next consider the maximum-likelihood estimate for the availability in the same setting, i.e., we again focus on a certain content type and the aim is to find the probability that a random node has a content matching a query of the given type (say, sport results). To this end, we assume similarly as in the previous section that

1. The number of queries originating from i hops away obeys a Poisson distribution, $M_i \sim \text{Poisson}(a)$, where a is an unknown parameter related to the rate of queries. (to be exact, $a = \lambda t$, where λ is the rate and t corresponds to the time-interval).
2. The number of observed queries corresponds to a thinned Poisson process, $N_i \sim \text{Poisson}((1-p)^{i-1}a)$.
3. The number of queries originating from different distances are independent, i.e., M_i and M_j are independent for $i \neq j$.

In other words, we assume that the mobile users have a common query rate λ, but they behave independently, which is a fair and common assumption (cf. telephone calls, web-sessions, etc.). With these, the likelihood function is

$$L = \prod_{i=1}^{H} \frac{\left(a(1-p)^{i-1}\right)^{n_i}}{n_i!} e^{-a(1-p)^{i-1}},$$

and, as usual, we can consider its logarithm,

$$\log L = \sum_{i=1}^{H} n_i \log(a(1-p)^{i-1}) - a(1-p)^{i-1},$$

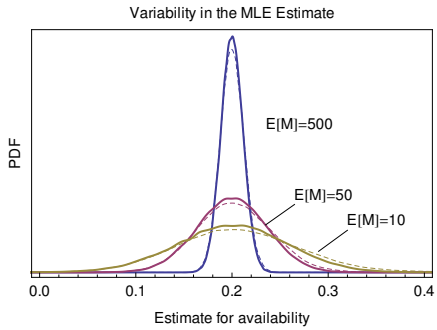


Figure 7: MLE based estimate (solid lines) yields a marginally better results than the regression based estimate (dashed lines).

where we have omitted the constant terms $-\log(n_i!)$. We have two unknown parameters, the mean number of queries per distance a, and the availability probability p, and we need to determine such (a, p) that maximizes the likelihood function. First,

$$\frac{\partial \log L}{\partial a} = \sum_{i=1}^{H} \frac{n_i}{a} - (1-p)^{i-1} = \frac{1}{a}\sum_{i=1}^{H} n_i - \sum_{i=1}^{H}(1-p)^{i-1},$$

and setting $\partial \log L/\partial a = 0$ gives

$$a = \frac{\sum_{i=1}^{H} n_i}{\sum_{i=1}^{H}(1-p)^{i-1}}. \tag{21}$$

Then taking the partial derivative with respect to p gives

$$\frac{\partial \log L}{\partial p} = \sum_{i=1}^{H} n_i \frac{a(i-1)(1-p)^{i-2}(-1)}{a(1-p)^{i-1}} - a(i-1)(1-p)^{i-2}(-1),$$

and further,

$$\frac{\partial \log L}{\partial p} = \sum_{i=1}^{H} -n_i \frac{i-1}{(1-p)} + a(i-1)(1-p)^{i-2}.$$

Setting also $\partial \log L/\partial p = 0$, and multiplying with $(1-p)$, we have

$$a = \frac{\sum_{i=1}^{H} n_i(i-1)}{\sum_{i=1}^{H}(i-1)(1-p)^{i-1}}. \tag{22}$$

Therefore, we are left to find such p that

$$\frac{\sum_{i=1}^{H}(1-p)^{i-1}}{\sum_{i=1}^{H} n_i} = \frac{\sum_{i=1}^{H}(i-1)(1-p)^{i-1}}{\sum_{i=1}^{H} n_i(i-1)},$$

from which a straightforward algebraic manipulation gives

$$\sum_{i=0}^{H-1}(c-i)q^i = 0, \qquad \text{where } c = \frac{\sum_{i=1}^{H} n_i(i-1)}{\sum_{i=1}^{H} n_i} \text{ and } q = 1-p. \tag{23}$$

Eq. (23) is a polynomial function of the $(H-1)$th degree. Hence, closed-form solutions are readily available for $H \leq 5$, whereas for polynomials of degree 5 or greater (i.e., when $H \geq 6$) a numerical solution is needed. Solving (23) gives an MLE estimate for p, which again when substituted into (21) or (22) gives a, i.e., the estimated number of queries originating from each distance. We refer to this approach as **Q-ML** as it relies on the query messages and uses maximum likelihood function for estimation.

Figure 7 illustrates the performance of the MLE based estimate (solid lines) in the same settings as before. The regression based estimate is also shown for comparison (dashed lines). We can see that MLE yields a marginally

better results (the peaks are higher and the right tail is lower), as one could expect. It is also more robust when the number of observed queries is small, as with about 0.2% probability the regression based approach failed to compute a meaningful estimate in the case $\mathrm{E}[M] = 10$. Moreover, the regression based approach seems to have a negligible bias and overestimates p slightly when $\mathrm{E}[M]$ is small. However, despite of these minor deficiencies, the regression based estimate is simpler to compute, and therefore we think that it is in overall a better option for actual implementations.

5.5. On availability estimation

Finally, we note that these methodologies are all local and do not require any additional communication between the nodes. They establish accurate estimates for the average content availability.[4] However, it is possible that nodes occasionally exchange their availability estimates, which increases communication costs only minimally. This can be extremely useful in the dynamic scenarios where nodes appear to the region and they can immediately get a good understanding of their surroundings. Another viable option is that nodes along the forward search path incorporate the availability information to the search process. That is, even if a totally new node initiates a search, already the first node along the search path can adjust the search parameters and include its availability information. In fact, the availability estimates can easily piggyback in the query and response messages without any additional communication costs. This is a topic for further research.

6. Performance evaluation

We now evaluate the performance of the availability estimation methods and the developed search strategies with several numerical examples. We start with the availability estimation and consider also search strategies where a node either has the complete information (e.g. a particular file) or nothing.

6.1. Availability estimation on Scale-free topologies

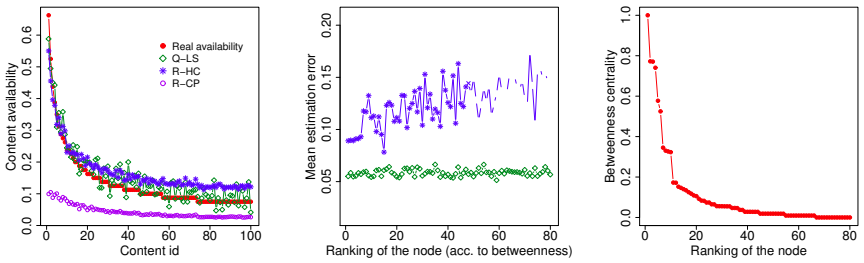
We consider a scale-free network of 80 nodes. Scale-free network [2] structure is prominent in real world networks, e.g., Internet, World Wide Web, human social networks. We simulate search on this network where all nodes generate queries for the contents in the network. We consider a uniform content popularity, i.e., all contents are equally likely to be sought. The content popularity in real networks is more asymmetrical, i.e., a few content items attract a significant fraction of the requests whereas there are only a number of requests for many content items. However, we consider a uniform popularity to avoid the highly popular contents being the dominant factor in the overall performance. Regarding availability distribution, we consider a skewed availability distribution. More particularly, content availability of an item is derived from a Zipf distribution with parameter 0.6, which reflects the Zipf parameter of some real world networks, e.g., Web proxy traffic [14]. While it is shown that content popularity follows Zipf distribution, there is not such wide agreement on content availability distribution. However, it would be realistic to expect that availability reflects the content popularity, i.e., if nodes —requesters and/or forwarders— cache response contents opportunistically [17], availability would grow with popularity. However, as discussed in [21] content availability may not always follow the same distribution as the content popularity. Nevertheless, such a skewed distribution serves to our purpose which is to see if difference in content availabilities can be captured by the estimation schemes.

Queries are routed using a flooding algorithm whereas the responses are routed back only on the shortest path from the content provider to the searching node. We report results as the average of five runs of each scenario. Since Q-ML is only marginally better compared to Q-LS but comes with a higher complexity, we omit it in this section.

In Section 5.3, we assumed uniform distribution for M_i. However, this assumption may not hold for real networks where nodes may have different number of nodes at a distance of i hops. We call the total number of nodes that are exactly i-hops away from a node as the i-hop neighborhood size and denote it by NN_i. Assuming that all nodes have similar interests and same query generation rates, we expect M_i to be proportional to NN_i. Hence, in our estimation using Q-LS, we assume that each node knows its NN_i values for $i = \{1, 2, \ldots, D\}$ where D is the network diameter.

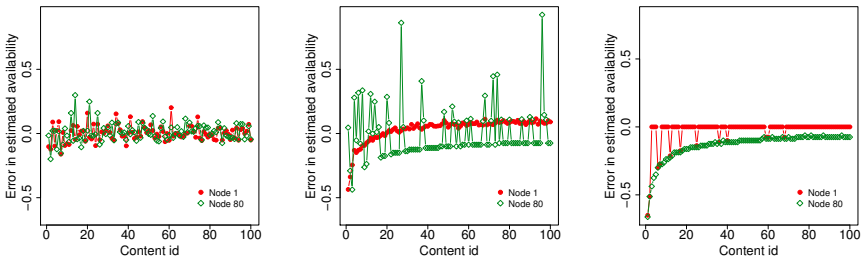
Fig. 8(a) illustrates the collective estimation for all schemes. While estimates may not exactly match the real availability, Q-LS and R-HC succeed in getting the skewed nature of the availability distribution. This result is appealing considering the simplistic nature of the estimation schemes. Ability to distinguish two content items according to their availability is of practical importance, especially in the realm of caching algorithms. Even if the estimated availability

[4] To be exact, these methods give estimates for the content availability in the node's own neighborhood, which is also the area that is relevant for node's own queries.



(a) Collective estimates. (b) Accuracy of the nodes. (c) Betweenness centrality.

Figure 8: Content availability estimation on a scale-free network.



(a) Q-LS. (b) R-HC. (c) R-CP.

Figure 9: Availability estimation errors for two nodes on a scale-free network.

diverges from the actual values with some error, Q-LS and R-HC can decide which item is more available out of a set of items. Using this information, a cache replacement algorithm can decide on which item to evict and which one to admit. Comparing the three approaches, Q-LS seems to have the highest accuracy which is closely followed by R-HC. R-CP has low accuracy particularly for highly available items. We attribute this to the fact that these requests are satisfied from very close nodes (e.g., one or two hops away) and therefore the responses are only relayed by a very few nodes. That means only very few nodes receive the response and improve their knowledge about the network. Besides, even if there are many content providers for a particular content, a node gets only one response from the closest provider. Therefore, only some (closest) providers are discovered by the nodes. Another thing to notice is that R-HC overestimates the availability almost all the time (except for the most available contents) whereas R-CP always (naturally) underestimates the availability. For the former, overestimation is due to the short network diameter of the considered network. In other words, the network has a small-world structure which is an emerging structure in human contact networks [4] and the longest path a query and response message can follow is much shorter compared to the number of nodes. For example, in the considered setting network diameter is 12 whereas average shortest path length is 5.45. Considering the availability calculation in (18), the calculated availability for a query following the longest path would be $1/12 = 0.08$, following a typical path would be $1/5.45 = 0.18$. Since the lowest content availability in our setting is 0.07 and average content availability is 0.14, R-HC results in overestimation of the availabilities.

Regarding the local estimates at each node, Fig. 8(b) suggests that Q-LS has small variance across the nodes whereas the approaches utilizing the response messages are markedly affected by the node's position in the network. From Fig. 8(b) and Fig. 8(c), we articulate that more central nodes have more observations due to their central position in routing messages and they can exploit the diversity in the message hop counts. This increased diversity gives less biased information about the content availabilities which in turn increases the estimation accuracy. Q-LS does not suffer from this effect as nodes do not solely rely on the number of messages with a particular hop count but rather consider the relative change in the number of received messages with increasing number of hop counts. To verify our claim, we calculated the correlation coefficient between the betweenness centrality of nodes and errors resulting in each approach. For Q-LS, correlation coefficient is -0.15 which can be interpreted as no correlation whereas it is -0.66 for R-HC and -0.43 for R-CP, which corroborates our claim.

Considering a distributed scheme where nodes do not exchange their estimations, we can state that the Q-LS is the best approach. However, it requires the knowledge of the neighborhood size of each node at a specific hop count.

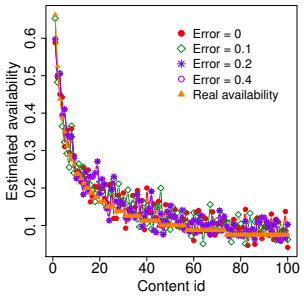


Figure 10: Estimated content availability under Q-LS with various error values in the knowledge of neighborhood size.

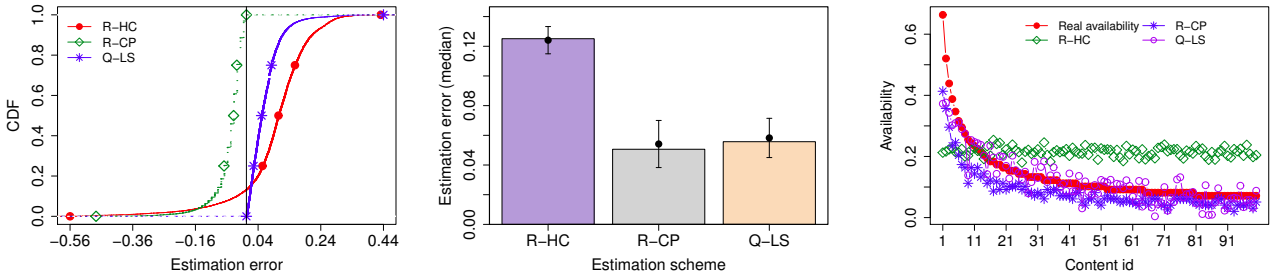
While this information could be gathered via message exchanges, it may not be always accurate. We depict the effect of inaccuracy in the knowledge of the neighborhood size. Let us define this inaccuracy as the fractional deviation from the actual neighborhood size. We consider the deviation from the actual values uniformly distributed in $[-\epsilon, +\epsilon]$. Fig. 10 illustrates the estimated availabilities under both perfect knowledge and imperfect knowledge: $\epsilon = \{0, 0.10, 0.20, 0.40\}$. All schemes maintain similar performance with $\delta_p \approx 0.03$.

From the previous figures, we infer only the absolute average errors but cannot observe how the estimation error changes across the content items. Let us take two nodes (n_1 and n_{80} with very different betweenness centralities) as example and analyze their estimation error for each content item under the three estimation schemes. Fig. 9 depicts the estimation error for these two nodes. Note that different than the previous figures, the estimation errors are signed where negative values stand for underestimation of the availability and positive values for the overestimation. The figures help us understand the difference in nodes' estimations as well as difference across the contents. As we already observed in Fig. 8(b), in case of Q-LS nodes exhibit similar performance in estimating the availability. Moreover, we do not observe a clear trend across the contents. In other words, estimation error does not depend on the availability of the items. In contrast, R-HC shows different behavior. From Fig. 9(b), we have two observations. First, n_1 has higher accuracy (i.e., errors approaching zero) compared to n_{80}. Second, although errors being quite small, n_1's estimations are mostly on the optimistic side, i.e., positive values except for the items with high availability. On the other hand, n_{80}, a node at the edge of the network, may both underestimate or overestimate depending on which queries and responses reach to this node. This effect is also observed in Fig. 9(c). As it is difficult to observe all providers of a content item, both nodes underestimate the availability. However, n_1 as the most central node maintains higher accuracy compared to n_{80} whose accuracy improves for contents with lower availability.

In a nutshell, Q-LS is the most accurate approach which is minimally affected by the node's structural position in the network as well as the content's availability. Regarding complexity, this approach requires the knowledge of neighborhood size. However, our experiments suggest that even when the nodes' knowledge of the neighborhood size is not perfect, nodes can acquire a good estimation of the availabilities using regression. Approaches using response messages - both R-HC and R-CP, are subject to variation depending on the node's centrality and the content's availability. However, they do not require any knowledge about the network. In terms of space complexity, given that opportunistic networks and many real world networks satify $H \ll N$, Q-LS and R-HC are less demanding with complexity of $O(H \times M)$ compared to R-CP of complexity $O(N \times M)$.

6.2. Availability estimation on human contact traces

In this section, we evaluate performance of our estimation schemes when various complexities of real world systems are in effect. We use `Infocom05` and `Infocom06` traces [20] that are collected from groups of conference participants carrying iMotes throughout the conference (i.e., approximately 3–4 days). `Infocom05` has 41 users whereas `Infocom06` has 98 nodes (including some stationary nodes). Every minute one of the nodes initiates a query for a particular content item. Admitted queries as well as responses for these queries are routed using hop-limited search protocol [5]. We set hop limit to 10 not to affect the estimations by hop-count based approaches. We consider Zipf availability with parameter 0.6 and Weibull content popularity distribution with $k = 0.513$ as suggested in [6]. We implement our scenarios using ONE simulator [15] and report estimations at the end of the simulation. For Q-LS, we derive estimations directly using (20). For R-HC, each node records exponential moving average of each content's availability based on the relayed messages. Estimated values are updated using equal weights for the previous estimation and the new observation.



(a) Distribution of estimation errors. (b) Average estimation error. (c) Collective estimates.

Figure 11: Results for `Infocom06` trace for each scheme.

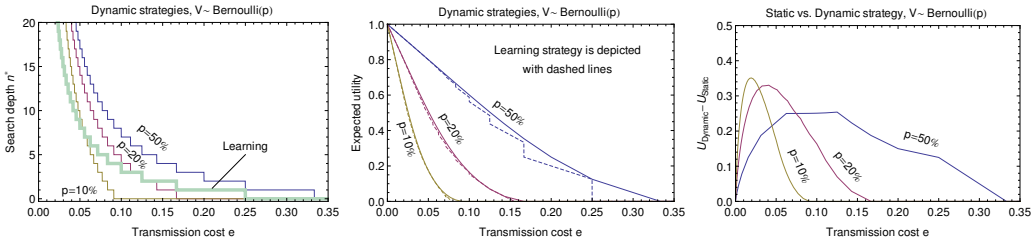


Figure 12: Left: The learning strategy unaware of p vs. dynamic strategies that know p exactly. Middle: Mean utility with the learning and dynamic strategies. Right: Decrease in utility if static strategy instead of the dynamic. ($\gamma = 1$)

Fig. 11 shows the results of our experiments on `Infocom06` trace. As key trends are very similar in `Infocom05`, we report results only for `Infocom06`. In Fig. 11(a), we see the distribution of (signed) estimation error. Values on the left of the vertical line at $x = 0$ stand for underestimations, whereas to the right values represent overestimated availabilities. While most of the schemes overestimate the availability, R-CP always underestimates as nodes may not observe all the content providers for a particular item. Fig. 11(b) plots the median of estimation error that is calculated over all nodes and all content items. Hop-count based R-HC significantly underperforms when compared to other two. This behaviour is due to the short diameter of the considered contact networks: the connectivity graph of the nodes at the end of the simulations is a full mesh. Similarly, query and response paths are very short, e.g., only a few hops. In Fig. 11(c), we plot the average estimated availability and the real availability for each content item. In agreement with Fig. 11(a), we observe the underestimations and overestimations of each scheme. Moreover, we can observe where the errors lie, either in the head or the tail of the skewed availability distribution. Note that the number of generated queries for content items follow a similar skewed curve, i.e., Weibull distribution. Therefore, nodes have different numbers of observations for each content: more observations for popular contents and only a few for the least popular one. However, for the considered setting we do not observe the effects of this divergence, possibly because of long observation time.

Regarding differences among nodes, `Infocom06` trace represents a well-connected network where all nodes have a similar contact patterns. Therefore, there is no significant differences between nodes' estimations.

In short, our evaluations show the feasibility of the considered estimation schemes under unideal practical settings. However, we should note that there are many aspects to be studied further, e.g., how to define accuracy metric, for more efficient estimation of availabilities.

6.3. Search of a specific content

Let us next consider different search strategies. Here 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. 12 (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 sought content, when $p = \{10\%, 20\%, 50\%\}$. Note

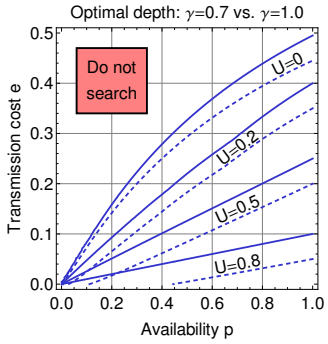


Figure 13: Equivalue curves for the utility U_{n^*} with the dynamic strategy when $\gamma = 1$ (solid lines) and $\gamma = 0.7$ (dashed lines).

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 sought information (normalized to one here) to the unit transmission cost e. We see that the learning strategy is an educated compromise.

Fig. 12 (middle) illustrates the resulting performance, i.e., the mean utility of a search in the three cases, $p = \{10\%, 20\%, 50\%\}$, as a function of the transmission cost e. 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. 12 (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. 13 illustrates the equivalue contours of the utility in the ideal case with $\gamma = 1$ (solid lines) against the setting where $\gamma = 0.7$ (dashed 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 ($\gamma = 0.7$).

7. Conclusions

Forwarding packets in a meaningful manner in opportunistic networks is a difficult task. The basic routing schemes such as the plain flooding and the *spray and wait* algorithm [22] 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 sought content must be delivered back to the searching node. The first 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. The second important contribution are the different methodologies to estimate the availability of the information *a priori* based on earlier queries each node observes. This information is valuable as it enables nodes to adjust the search parameters optimally. In more general terms, the question is what the query can learn from the surroundings, and how to capitalize that information. Other possible quantities that might be exploited are the node degrees (with an appropriate definition in this possibly highly dynamic context), and community memberships.

In many cases, we were able to characterize the optimal search strategy that maximizes the expected utility with the given initial information. 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. Similarly, the heterogeneity of the actual scenarios poses a notable challenge.

Finally, we introduced several schemes for availability estimation and discussed the pros and cons of each approach. While accuracy of each scheme depends on the internals of routing, e.g., how many hops, search stopping criteria, TTL of the messages, we presented both analytical explanation and simulations for evaluating the performance of the proposed techniques.

Acknowledgements

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

[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] Albert-László Barabási. Scale-free networks: a decade and beyond. *Science*, 325(5939):412, 2009.

[3] 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.

[4] Suzan Bayhan, Esa Hyytiä, Jussi Kangasharju, and Jorg Ott. Analysis of hop limit in opportunistic networks by static and time-aggregated graphs. In *IEEE International Conference on Communications (ICC)*, 2015.

[5] Suzan Bayhan, Esa Hyytiä, Jussi Kangasharju, and Jorg Ott. Two hops or more: On the hop limit in opportunistic networks. In *ACM MsWiM*, 2015.

[6] Xu Cheng, Cameron Dale, and Jiangchuan Liu. Statistics and social network of youtube videos. In *Int. Workshop on Quality of Service (IWQoS)*, 2008.

[7] 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.

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

[9] 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.

[10] Jialu Fan, Jiming Chen, Yuan Du, Ping Wang, and Youxian Sun. Delque: A socially aware delegation query scheme in delay-tolerant networks. *IEEE Transactions on Vehicular Technology*, 60(5):2181–2193, Jun 2011.

[11] 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.

[12] 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.

[13] Esa Hyytiä, Suzan Bayhan, Jörg Ott, and Jussi Kangasharju. Searching a needle in (linear) opportunistic networks. In *Proc. of The 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, 2014.

[14] Jussi Kangasharju, James Roberts, and Keith W Ross. Object replication strategies in content distribution networks. *Computer Communications*, 25(4):376–383, 2002.

[15] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The one simulator for dtn protocol evaluation. In *Proceedings of the 2Nd International Conference on Simulation Tools and Techniques*, Simutools '09, pages 55:1–55:10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[16] 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.

[17] Jörg Ott and Mikko Pitkänen. DTN-based Content Storage and Retrieval. In *The First IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC)*, June 2007.

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

[19] 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.

[20] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. CRAWDAD dataset cambridge/haggle (v. 2009-05-29). Downloaded from http://crawdad.org/cambridge/haggle/20090529, May 2009.

[21] Pavlos Sermpezis and Thrasyvoulos Spyropoulos. Not all content is created equal: Effect of popularity and availability for content-centric opportunistic networking. In *MobiHoc 2014*. ACM, 2014.

[22] 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.

[23] Hongyi Wu, Yanyan Han, and Zhipeng Yang. Efficient data query in intermittently-connected mobile ad hoc social networks. *IEEE Transactions on Parallel and Distributed Systems*, 2014.

[24] 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.