Searching for Content in Mobile DTNs

Mikko Pitkänen[1], Teemu Kärkkäinen[2], Janico Greifenberg[3], and Jörg Ott[2]
[1]Helsinki Institute of Physics, Technology Programme <mikko.pitkanen@hip.fi>
[2]Helsinki University of Technology, Department of Communications and Networking
[3]Dampsoft GmbH

***Abstract*—Delay-tolerant Networking (DTN) provides a platform for applications in environments where end-to-end paths may be highly unreliable or do not exist at all. In many applications such as distributed wikis or photo sharing, users need to be able to find content even when they do not know an unambiguous identifier. In order do bring these applications to the domain of DTNs, a search scheme is required that works despite the unreliable network conditions. In this paper, we introduce a search scheme that makes no assumptions about the underlying routing protocols and the format of search requests. We evaluate different algorithms for forwarding and terminating search queries, using simulations with different classes of DTN routing protocols for different mobility scenarios.**

I. Introduction

Retrieving information from the Web has become part of our daily lives. Users often have an idea what content they want to retrieve, but do not know a priori an exact identifier (or locator) to feed into an application for the resource retrieval. A search engine helps by mapping search queries to resource locators that are likely to point to content including the sought information. However, there are scenarios in which communicating with a search engine is not an appealing or even suitable option. Such situations occur, for example, when the desired information may be available from the vicinity via a local wireless (ad-hoc) network and a connection to the search engine is unavailable or not reliable. In this case, a search mechanism is desirable that allows directly querying other nodes in an unreliable networking environment.

The traditional approach to map search queries to resources is implemented by maintaining an index about the resources. The index contains keywords and maps them to sets of related resources. This can be used to map a query consisting of several keywords to a list of suggested search results. Examples of centralized search engines using such indices are Google, MSN, and Yahoo. The problem with such centralized services is that it is expensive to keep the index up-to-date and the assumption is made that the clients can reach both the index and the locations where the contents reside. There are approaches for managing decentralized indexes in peer-to-peer environments, but these models often assume good connectivity among the nodes. Otherwise, unreliable connections may lead to a high maintenance cost of the index.

Furthermore, not all contents of interest resides on servers connected to the infrastructure network: mobile users are not just consumers, but also prime producers of contents and often eager to share their impressions with others. Particularly in disconnected environments, this "mobile" contents cannot easily be indexed by centralized search engines unless uploaded to a some server. Yet, such contents may be especially interesting to other mobile users in proximity, which could access it by means of ad-hoc networking.

In this paper, we investigate how to support searching and retrieval of contents in unstructured delay-tolerant mobile ad-hoc networks. We choose delay-tolerant networks as traditional ad-hoc networking places tight demands on connectedness of the nodes, which appears infeasible in our scenario. This paper extends our earlier work on opportunistic content caching and retrieval using resource identifiers [17] [18] and complements related work on using DTNs for active content dissemination to interested users [16] [14]. We introduce a simple notion of content queries based upon which we present how nodes should process and forward queries and respond to them returning matching locally available contents. Nodes carry (possibly self-produced) content items (in the context of the web also referred to as *resources*) which can be matched by a local search function against incoming queries generated by other nodes. They may respond to and/or forward the query to one or more other nodes. When responding, a node may choose to return all or a subset of the matching local contents. We restrict our considerations to searching the mobile ad-hoc environment and do not discuss interaction with infrastructure networks.

After reviewing related work on search from various fields in section II, we discuss the search problem for DTNs in section III where we also present different search strategies. We introduce metrics for search performance and evaluate the different search strategies in section IV. We finally present considerations on our prototype implementation in section V and conclude this paper with a brief assessment of our results and hints at future work in section VI .

II. Related Work

Searching for information is an area of active research in the computer science community with a wide range of publication with specializations from algorithms and operations of large scale databases to search in various application scenarios. As this paper is focused on distributing search requests and results in challenged networks rather than the process of actually finding the results, we limit the discussion of related work to efforts similar to ours that perform search in opportunistic networks and to studies about user behavior which give us insights about requirements.

A. Search in Unstructured and Ad-hoc Networks

There are two basic approaches for searching in unstructured networks: flooding and random walk. Most proposed flooding-based schemes use a TTL-based limit to control the spread of the queries, e.g. as described in [6]. In contrast, search-schemes based on random walks [2], [20], avoid the massive spreading of messages that flooding creates and still achieve a degree of reliability by using probabilistic paths to reach responders.

BubbleStorm [24] is a probabilistic search strategy in peer-to-peer networks. It is based on a combination of replication and probabilistic distribution of queries. As BubbleStorm depends on a network that is a random multigraph with a fixed degree distribution, the system is not applicable for DTNs, as nodes cannot control the number of neighbors to peer with as peer-to-peer systems working over the Internet can.

Adamic et al. [1] present search strategies that exploit power-law degree distributions. Their method passes a search request from one node to another, choosing the neighbor with the highest degree. After the node with the maximum degree has been reached, it will be avoided, thus the search descends in the degree sequence.

Yang and Hurson [26] present probabilistic schemes for locating content in wireless ad-hoc networks. Based on knowledge about the query history, they use heuristics and Bayesian probability calculations guide the dissemination of queries.

Hui et al. have presented a search mechanism for pocket switched networks (PSN) called *Osmosis* that derives from analogy to its biological counterpart [9]. In Osmosis, the queries in the network are spread based on epidemic forwarding, and the results are routed back based on the traces that queries left while traveling between requesting node and the responding node. In our work, in contrast, we limit the system resources used to spread and evaluate searches and rely on different existing routing mechanisms to deliver responses.

Balasubramanian [3] et al. have presented a system called *Thedu* to enable efficient web search from a city bus. Thedu acts as a web proxy and collects search queries from a mobile user at the time of disconnections. Thedu differs from our work by assuming that nodes can eventually establish a connection to the Internet where a centralized search engine can be accessed. Our proposal, on the other hand, allows search in completely disconnected networks, where all content is distributed among the nodes.

Several studies exists on how to fetch resources in DTNs identified by a previously known unique resource identifier (URI). Earlier work by the authors has investigated web resource retrieval in mobile DTNs via Internet gateways [17] and shown how caching improves efficiency with shared interest on resources [18], using well-known DTN routing mechanisms.

B. Search Usage Studies

There are a number of studies analyzing the use of web search. Based on the logs of the Excite search engine for one day (in March 1997) Jansen et al. [11] found that the majority of web searches consist of only few terms, and that the frequencies of occurrences of search terms exhibit a highly skewed distribution with 44% of the search terms occurring only once. Search sessions—i.e. a series of queries by a single user within a short period of time—were also found to be short.

A similar study by Silverstein et al. [21] based on AltaVista query logs from 1998 supports the results of [11]. A later study [12] using Altavista data from 2002 showed a slightly higher number of terms per search and the diversity of search terms was found to be larger than in previous studies with the most frequently used term accounting for only 0.6% of all queries.

Kamvar and Baluja [13] analyze wireless search behavior based on Google's mobile search logs gathered during one month. Their results are in line with the general properties observed in the studies cited above whose data is mostly about searches from desktop computers: short queries and sessions and a high variation of the search terms. However, there is an observable difference between the types of devices. Searches from cellphones are slightly shorter than those from PDAs. The distribution of the used search terms is more biased for cellphones where the top 1000 queries account for 22% of all queries than for PDAs where the top 1000 represent only 6% of all queries.

Church et al. [7] present an extensive study of mobile information access conducted in 2005 based on logs from mobile a provider which include both search and browsing patterns. Their results are in line with the findings of the analysis based on Google mobile logs, and they predict that the diversity of search interests will grow for mobile search as it did for web search in general.

In summary, we see that searches are in fact repetitive, especially in the mobile space, but the variety of topics is very large, so that the top searches constitute only a fraction of the total search traffic. As a consequence, optimizing based on the most popular searches will have only a limited effect on the overall user experience. Unfortunately, none of these studies provides insight on the similarity of the search results. With the available numbers, we can only infer that different queries also matched different results, although we cannot tell to what extent this is true.

III. Searching in Delay-tolerant Networks

In this section we analyze the problem of search in delay-tolerant networks and describe alternative solutions which are evaluated in section IV. As stated in the introduction, we focus on mobile opportunistic networking scenarios where the nodes communicate using the DTN bundle protocol [5], [19]. Some devices in the network store content which they are willing to share with others. All nodes are willing to cooperate and supply a limited amount of their local system resources (bandwidth, storage, processing power) to assist other nodes.[1] Our goal is to allow users to issue queries for content that is stored on other nodes anywhere in the network and assess the

[1] We defer considerations of fairness and defense against malicious or selfish nodes to future work.

chances of such a node to obtain the sought information. To facilitate searching, we assume that nodes are able to perform searches on their local storage and find the relevant results for a given query.

We define N as the set of nodes, C as the set of all content items that are available in the network, and Q as the set of queries issued over the lifetime of the network. To identify on which nodes a given content item is stored, we define the function cl that maps a content item to a set of nodes: $cl(c) \subseteq N$, for $c \in C$. The mapping from a node to its stored content items is defined by $s(n) \subseteq C$, for $n \in N$. We also define the function m that maps queries to the content items they match: $m(q) \subseteq C$, for $q \in Q$. $size(c) \in \mathbb{N}$ yields the size in bytes of $c \in C$. Finally, we define $rsp_{local}(q, n) = m(q) \bigcap s(n)$ to be the response generated by a node n when replying to query q and the function rsp that yields the responses for a query seen by a given node: $rsp(q, n) \subseteq m(q)$, for $q \in Q$ and $n \in N$. This includes the locally generated responses.

Conceptually, we divide the processing of a query into three phases: (A) During *local matching* phase nodes execute the query searching for content in their local storage and generate a set of all matching resources, from which the *selection* function picks a subset to return. Partly as a function of whether or not results were found, (B) the node runs a *query processing* function that decides about forwarding the query or about search termination, as well as modifying the query. Finally, (C) *response forwarding* is responsible for routing and forwarding locally generated responses and those received from other nodes.

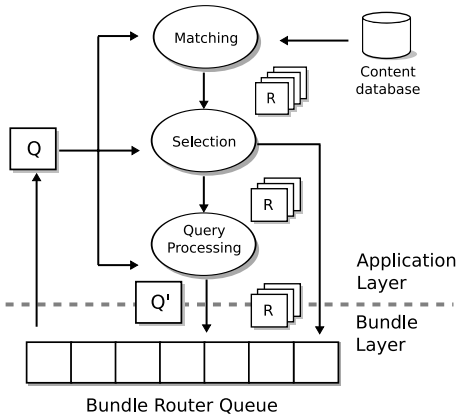

Fig. 1. Processing flow for an incoming query

Figure 1 illustrates the node's query processing flow. When a query arrives at a node, the matching logic retrieves all content items by executing the search on the local content database. The matching resources along with the original query are passed to the selection logic which selects a subset of the content items to return. This decision can take into account information carried by the query, such as the number of resources returned by other nodes. Finally, the resources and the query are passed to the query termination logic which chooses to terminate or forward the query. The original query may be modified by appending information, such as the number of returned resources, before being passed to the bundle layer for forwarding.

The simplest strategy for distributing both queries and responses is flooding. However, due to resource limitations, flooding is often too wasteful so that we need to consider methods to limit the system resources spent on a search. Ideally, a query should only be forwarded to neighbors that hold matching contents or are on the path to other nodes which do and different nodes should return non-overlapping responses to the searcher. As global knowledge or active coordination is not an option in our scenario, each node can only select next hops for queries in such a way that the system resources used on a search are balanced against the probability that it yields useful results. Similar limitations apply to local content selection and routing responses.

In the following subsections, we discuss effective controlling of response matching and selection, query forwarding and termination, and response forwarding. In this paper our emphasis is on the *query forwarding and termination* part and we will address the other two only briefly. We focus on strategies for limiting the number of copies of a query that are distributed and the number of responses to be returned. It is concievable, however, to specify the limits in greater detail. Instead of just limiting the number of copies to be distributed for a query, the sender could specify how many CPU cycles are to be used to evaluate the search and how many bytes of results are to be returned. This can help to delimit both the consumed storage space and the required bandwidth. Analyzing these kinds of limits further is beyond the scope of this paper.

A. Local Matching and Selection

A wide variety of systems to formulate search queries exist, most of which depend on assumptions about the format of the content to be searched. While it is desirable for a general protocol to support many different types of queries and contents, for the purpose of this work, we abstract from the actual matching process to focus on what is transmitted over the network. In our evaluation, we use a simple keyword matching to substitute for the actual execution of queries.

When a node has found the set of matching content in its local storage, it may wish to respond with only a subset of the results. This can be done to limit the amount of resources used both locally and globally for transmitting and storing the responses, or to weed out potential duplicates.

The originator may assign a boundary for either the number of responses or the volume of response-data that should be generated. This limit gets encoded in the query and each node then subtracts the number of results found in its local storage and never returns more than the allowed number of responses. Before forwarding, the respective field in the query message is adapted accordingly.

If the local matches exceed the reply limit, the node needs to select which ones to forward. When the matching process yields a score for the relevance of the results (in our simple case, e.g., the number of matching attributes), we can use this

to choose the elements of the selection set. If this is insufficient (e.g., if only a single attribute was used in the query), the resources to be returned are chosen at random so that the chance for the searcher receiving only duplicates is minimized.

B. Query Processing

For the query distribution phase, we examine four alternatives for limiting the spreading of queries: a *hop-count limit*, a limited *time-to-live* (TTL), a simple *first matching response* drop, and a termination based on *global response count estimate* with information recorded along the path of the query message. The first two approaches can be implemented completely at the bundle layer without any input from the application layer. First response drop and global response count estimates require application layer logic but the former is based on explicitly known information while the latter requires estimation of a number of parameters.

When using a hop-count limit, the query is discarded once it has been forwarded L times. The limit L can be chosen by the originator of the query or it can be selected by the network, even by a node specific policy in which case the hop-count limit on different paths may vary. Denoting the number of hops a query has traveled as $hc(q)$, the forwarding condition for query q in node n with a hop-count limit can be formalized as:

$$forward(q,n) = \begin{cases} true & \text{if } hc(q) < L \\ false & \text{otherwise} \end{cases}$$

The time-to-live limit $ttl(q)$ of a query q is a similar approach in which the query is forwarded for a period of time specified by the originator, independent of the number of nodes visited. Under this scheme the spread of queries is likely to differ significantly depending on the node density of the scenario. With t_{now} as the current time and $t_{creation}(q)$ being the time when the query q was created, we define the following forwarding condition:

$$forward(q,n) = \begin{cases} true & \text{if } (t_{now} - t_{creation}(q)) < ttl(q) \\ false & \text{otherwise} \end{cases}$$

When using a single-copy routing scheme such as first contact, the spread of queries with both hop-count and TTL limits is essentially a random-walk, where the next-steps are determined by the motion of the nodes and the rules of the routing scheme. When used with replication-based routing, the queries are distributed in the form of controlled flooding (based upon which expanding ring searches could be realized).

The first response drop scheme forwards the query if no local match was found and terminates the search otherwise; i.e., a dependency on the local contents is introduced which impacts DTN routing.

$$forward(q,n) = \begin{cases} true & \text{if } (rsp_{local}(q,n) = \emptyset) \\ false & \text{otherwise} \end{cases}$$

Finally, we formulate a search termination function that takes into account further information recorded along the path of the query message. While two-way coordination between nodes is not feasible in our scenarios, it is possible to pass information one-way along with the query message. The bundle layer is assumed to pass hop-count and transit time information to the application layer, while the response count can only be recorded at the application layer.

The goal of this search termination mechanism is to limit the spread of the query as the number of responses grows. In other words we assume that the marginal value of returning additional responses to the originator of the query decreases when the total number of responses increases. This can be justified by at least two arguments: 1) the likelihood of additional responses being unique decreases (due to the birthday paradox), and 2) the value of additional unique responses decreases (the user might have already received an acceptable response).

As each node has only a limited view of the network, it needs to estimate the number of responses that the query originator has already received. In order to make this estimation the node can guess the number of nodes that have seen this query globally and the number of responses each of those nodes has generated. For this, we define the function $u_{est}(q,n)$ that is directly proportional the number of responses generated globally to the query q based on knowledge at the node n.

Once we have constructed $u_{est}(q,n)$ we can use it to terminate the query when its value exceeds some threshold value T. This can be stated as:

$$forward(q,n) = \begin{cases} true & \text{if } u_{est}(q,n) > T \\ false & \text{otherwise} \end{cases}$$

In order to construct $u_{est}(q,n)$, we need to estimate the number of nodes that have received the query. We use a hop count hc recorded in the query which is initialized to 0 at the querying node and incremented at each hop to determine the number of nodes a query has already traversed. From this, the number of nodes $nodes_{est}(q,n)$ potentially having received a copy of the query after up to hc hops can be estimated as a function of the routing protocol. For single-copy routing protocols, $nodes_{est}(q,n) = hc(n)$. For multi-copy routing protocols with a fixed number of copies K, $nodes_{est}(q,n) = K - K(q)$, with $K(q)$ indicating the number of copies the received message represents. For epidemic routing protocols, every node on the path records its own *node degree* estimate ($degree(n)$), which it calculates as a moving average of the number of contacts it had during the last $ttl(q)$ seconds. This represents the possible replication factor at this node for the query, yielding a local approximation of the number of nodes reached by the query can be defined as:

$$nodes_{est}(q,n) = degree(n)^{hc(q)}$$

We can further estimate the number of responses generated per node by recording in the query message the total number of responses, $rsp_{tot}(q)$, generated by nodes along the path. If no responses have been generated along the path, a static estimate C can be used instead. This estimate can be either pre-configured or calculated as an average from past queries. Using the above, we approximate the global number of responses as:

$$u_{est}(q,n) = \begin{cases} nodes_{est}(q,n) \cdot \frac{rsp_{tot}}{hc(q)} & \text{if } rsp_{tot}(q) > 0 \\ nodes_{est}(q,n) \cdot C & \text{otherwise} \end{cases}$$

There are a number of assumptions in the above approach that should be considered. First, it is assumed that the global node degree distribution has an existing mean and that the node's measurement of its own degree is a good estimate for the average degree in the network. Should the node degrees be, e.g., power-law-distributed, the nodes that have very large node degrees will drastically overestimate the number of nodes that the query has been distributed to and are therefore less likely to forward the query. Second, since $node_{est}(q,n)$ is dependent on the routing protocol the application layer must be aware of the underlying routing and the routing must be uniform across the whole network. Nevertheless, we consider the above to be a fair estimate based on the limited knowledge available to the nodes.

C. Response Forwarding

When a node has found matches for a search request, it needs to transfer the results to the searching node. We do not define any search-specific functionality here but, instead, simply rely on the underlying routing protocol to deliver the reply messages to the originator of the query. This may lead to two types of duplicate responses:

1) When using multi-copy routing protocols, even replies from a single responder may get duplicated and reach the querying node via different paths (even though such routing protocols typically suppress forwarding duplicates in every single node). This is inherent to the nature of the routing protocol (and a perfect one would not generate any duplicates in the first place), so that further optimizations would need to further improve the routing protocols themselves—in conjunction with or independent of searching.

2) Identical or partly overlapping replies from multiple responders may reach the searching node. The above response selection mechanism attempts to reduce such duplicates heuristically and the search termination mechanisms helps by limiting spreading the query. However, we cannot prevent identical or overlapping responses from being generated by different responding nodes. This duplication occurs at the application layer and only there the message semantics (i.e., being identical/overlapping) are known. At the DTN routing level, messages are typically identified by the pair of originator and destination address and a locally unique identifier assigned by the originator so that messages from different responders appear as different ones to the DTN routing and are hence not suppressed as duplicates. Extensions are conceivable to explicitly tag messages specifying their content and the query they respond to in a way visible to a partly *application-aware* routing layer, e.g., in the message's metadata as partly suggested in [23], [17]. This could aid application-aware nodes in suppressing semantically duplicate messages.

We do not consider any such optimizations further in this paper to avoid too close an interaction with the routing layer (the detailed analysis of which would be well beyond the scope of this paper). Also, note that limited numbers of duplicates are not necessarily negative in DTN routing as they may increase the delivery probability—which is the reason for pursuing multi-copy routing in DTNs in the first place.

IV. Evaluation

For our evaluation, we use the Opportunistic Networking Environment (ONE) simulator [15]. We run simulations using four different network scenarios, including an easy to understand static scenario and different mobility models as described below. For all figures we plot average result of five simulation runs. The nodes comprising the network constitute devices of mobile users which are able to connect to each other by using bi-directional links at 2 Mbit/s. Communication between nodes is based on the DTN store-carry-and-forward networking model; the nodes have unlimited FIFO message queues for DTN routing. Messages are transmitted in an all or nothing fashion, i.e., fragmentation is not allowed.

We experiment with the four strategies for limiting the spread of messages in the network described in section III, with varying time-to-live and hop count constraints and value thresholds. While the query forwarding and search termination algorithms are designed to operate independently of the routing protocol, the routing protocols may create different numbers of messages in the network. For our evaluation, we choose three different routing protocols representative of different types of message replication. *First Contact* routing [10] represents the class of single-copy routing strategies, the other two use multi-copy behavior: *Spray-and-Wait* [22] limits the number of copies to a fixed maximum (we use 10 copies) and *epidemic* routing [25] performs flooding without any limitation on message replication.

The content query messages are sent to a DTN multicast address (*all-dtn-search-routers*) which yields anycast-style semantics in our search environment. Every (search-capable) DTN node joins this multicast address and is thus notified about an arriving query message. While unicast DTN messages will be deleted after delivery, in our scheme all nodes are intended recipients and messages may thus be locally evaluated by the search engine of multiple nodes, even if only a single copy is created and passed around. Besides adding local message delivery, we do not change the forwarding rules of the routing protocols; i.e., First Contact only uses a single message and Spray-and-Wait will deliver the message to at most 10 nodes.

For searches limited by hop count and time-to-live, the respective search termination leverages our regular DTN routing implementation without any additions. For first response filtering and value-based termination, a special extension to DTN routing is introduced: message forwarding may be deferred until the respective search application on the node signals either *forwarding* consent, possibly updating the query contents, or deletes the message (*termination*).

Response messages are addressed to the querying nodes and sent using the underlying routing scheme without any changes.

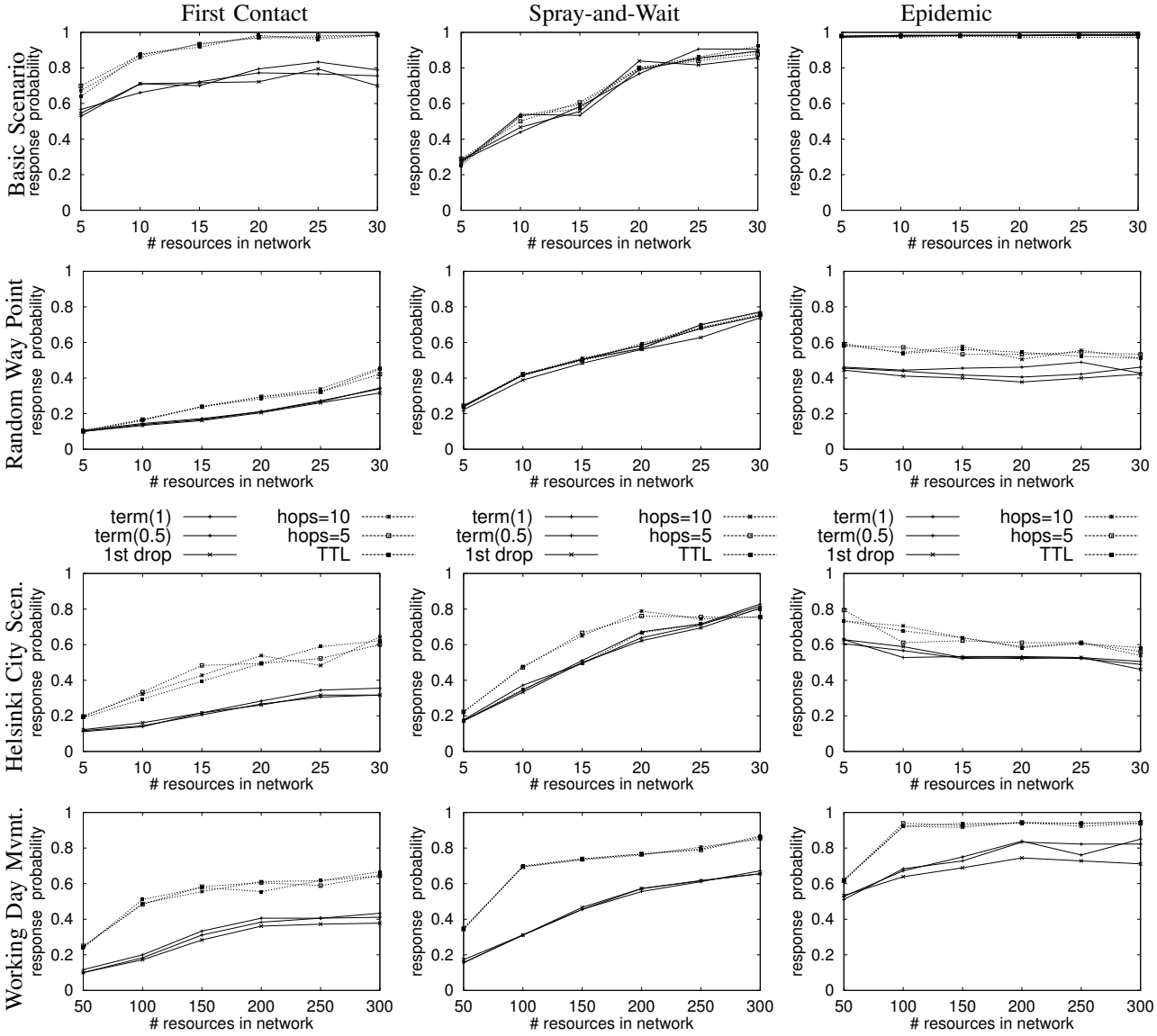

Fig. 2. Response probability for search queries.

In the following simulations we focus on investigating searching of a single content item from a mobile DTN and then extend our considerations to searches which can yield multiple distinct items. We observe searching the network from the point of view of a single actor. The actor frequently issues queries to the surrounding nodes, which carry a fixed and predetermined set of resources. These nodes respond by sending out contents matching the query in a reply message. We distribute resources prior to starting simulations so that a fixed number of nodes are able to provide a response. We then vary the number of responding nodes (represented on the x-axis) and plot the observed metrics as a function of number of potential responses. While typical scenarios could see many nodes initiating searches simultaneously, we deliberately constrain ourselves to a single requester as we want to isolate the performance of the search distribution from effects due to congestion.

For each query, the network contains a number of distinct, matching content items. Copies of the content items are distributed in the nodes' data stores in such a way that their popularity follows the Zipf distribution, which properly models the popularity of web content items [4] and is thus a reasonable assumption. The total number of copies of content items is fixed for each simulation run. For the first three scenarios the number of copies of the most popular content item is half the total number of copies, for the fourth (which features more nodes), it is 20% of the total. The size of the content items is fixed at one megabyte while the size of the query messages is insignificant in comparison.

A. Simulation Scenarios

Basic Scenario (BS) is a static network topology with a central node around which other nodes are spread in concentric circles. The distance between the circles is 100 units, each circle having six more nodes than the next inner one. With a node transmission range of 150 units this results in node degrees between three and six for all the nodes in the network. We also generated similar simple networks with each node having the same degree between four and eight. The search query is always issued at the middle node of the network.

Random Way Point (RWP) presents a well known case for comparison. Our model contains 125 actors walking in an area of 100×100 meters, comparable to an outdoor exhibition. The walking speed varies in range $0,5-1m/s$.

Helsinki City Scenario (HCS) [15] is based on simulating 80 mobile users moving by foot, 40 by car, and 6 by trams in the streets in downtown Helsinki. Each node represents a user moving with realistic speed along the shortest paths between different points of interest (POIs) and random locations. The nodes are divided into four different groups having different POIs and different, pre-determined probabilities to choose a next group-specific POI or a random place to visit. The trams follow real tram routes in Helsinki.

Working Day Movement (WDM) is used to simulate a more realistic scenario whose characteristics such as inter-contact time distributions come close to those encountered in real-world traces [8]. We choose the default scenario from section 5 in [8] in which nodes move in the Helsinki city area, but reduce the number of nodes from 1029 to 544 by shrinking all the group sizes so that the basic contact characteristics remain.

B. Simulation Observations

Figure 2 shows the probability for retrieving a single specific content item in response to a search request issued by a node. We investigate value thresholds of 0.5 and 1 (referred to as $term(0.5)$ and $term(1)$), first response drop, hopcount limits of 5 and 10, and a TTL limit matching the respective scenarios (BS: 10min, all others 120min). As expected, the success of a search increases when the number of resources in the network increases. Both, first contact and spray-and-wait routing show poor performance when there is only small number of potential responders in the network. Obviously, the small number of copies (or short route of a single copy) for a query does not lead to a high likelihood of reaching a node carrying a matching resource. Epidemic routing performs well with all mobility models and is less dependent on the resource density. This is an expected result when there is no contention and the query and its responses can reach large number of nodes.

We observe that the query forwarding and termination mechanisms tend to split into two groups: termination mechanisms which are based on network level control (TTL, hopcount) and those which utilize application level information (local information as first response drop and signaled information as value-based). The former consistently yield higher retrieval success rates. With WDM mobility, the network level control leads to significant improvements whereas, with the other approaches, the improvement over local control (first response drop, marginal valued function u_{est}) is less pronounced to non-existent. This holds irrespective of the number of resources spread across the network.

Figure 3 shows the mean number of unique replies per resource received in response to a query by a single querying node in the HCS scenario.[2] In this case multiple resources match a query. The resources are distributed randomly, the number of copies per resource following a Zipf distribution. In this figure, we show only one value-based termination (threshold 0.5) and only the TTL as representative for network level control: TTL and both hopcount limits perform almost identically and so do both value thresholds, which seems to indicate that expanding the search coverage by itself does not increase performance for these scenarios. The results obtained across all routing protocols reflect the occurrence frequency of the content items. However, we find that using first contact or spray-and-wait may not obtain even the most widespread content item in all cases. In contrast, epidemic routing expectedly provides broader coverage with repeated responses for the most frequent resources. Counting also replicated responses created by the routing protocol further shows that epidemic routing adds robustness by returning tenfold number of responses. With spray-and-wait the replicated responses lead to twofold increase in number of delivered responses. Limiting the query forwarding (and thus the resource consumption) with application layer knowledge is difficult; particularly, first response drop is not effective. TTL limiting is a workable limiting mechanisms and still allows even less popular resources to reach the querying node. Overall, we find that queries will likely be satisfied only for the more popular resources; content items from the long tail will only be captured occasionally and most likely from the vicinity of the querying node as our results for the hop count of the returned responses seem to indicate. This also means that exhaustive searches for content items in mobile DTNs are unlikely to succeed (within a limited time frame) and should be avoided as they may be wasteful.

Figure 4 shows the spread of query messages in the network with epidemic routing to further assess the effectiveness of the limitation functions (which is also a metric for the incurred load on the network).[3] We can see how the network level mechanisms (hopcount, TTL) cause widespread forwarding of the messages (reflected in the better retrieval performance as observed above). In the simple scenario, the first response drop mechanism manages to limit the spread of queries especially when large number of content items are in the network, and thus close to the querying node. With the (less realistic) RWP network level mechanisms limit queries as effectively as application level mechanism but achieve good response performance. All application level termination mechanism limit the spread of searches in HCS effectively and still manage to

[2]Note the threefold difference in range of the Y axis for Epidemic compared to First Contact and Spray-and-Wait.

[3]For First Contact (hopcount=10) and Spray-and-Wait (10 copies), the maximum spread of the messages has a hard limit enforced by the routing protocol and the graphs are omitted.

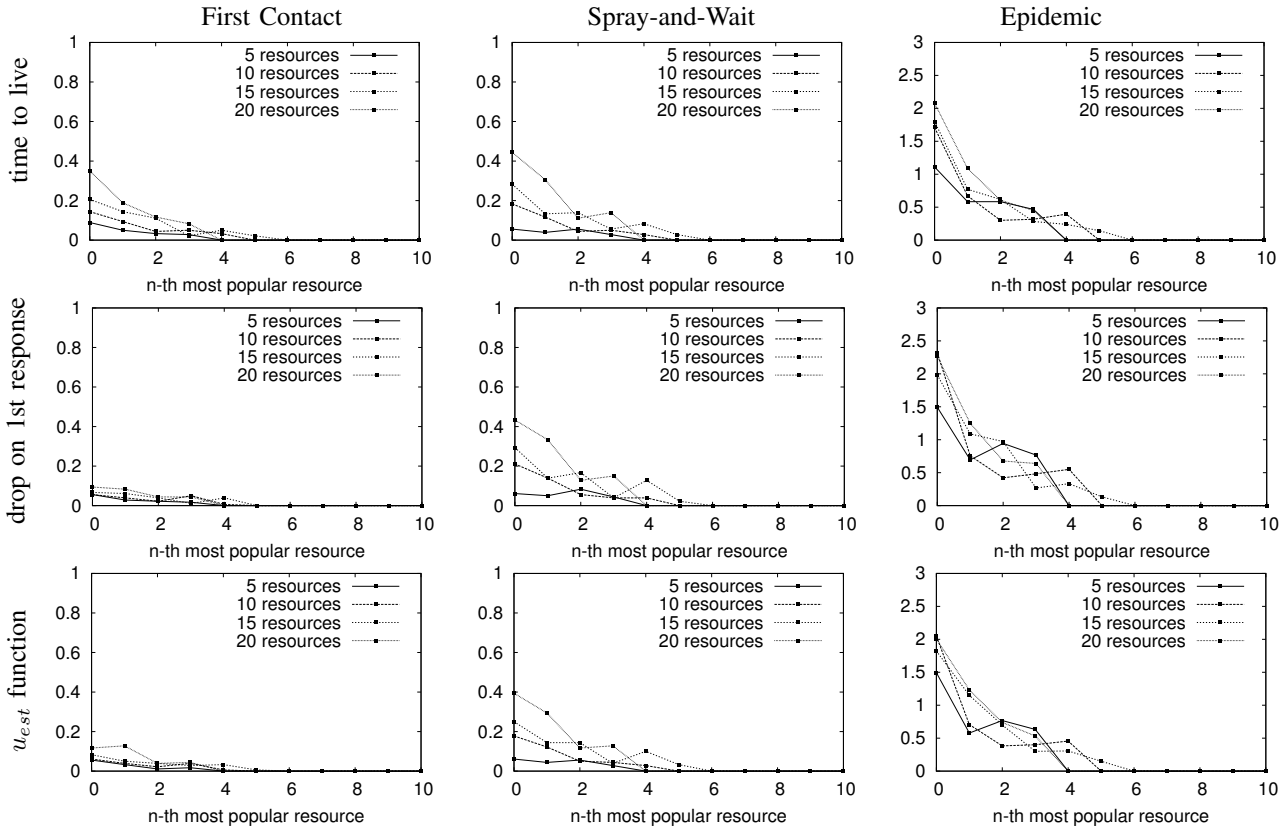

Fig. 3. Simulation results for Helsinki City Scenario.

deliver responses fairly well as can be seen in Figure 2 for the Helsinki City Scenario with epidemic routing. In WDM model, all limiting mechanisms stop queries equally well but the network level mechanisms lead to a better response rate as observed above.

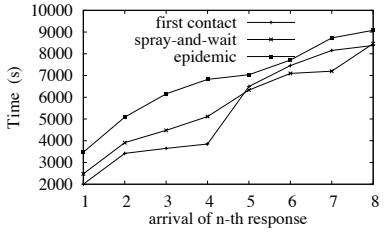

Fig. 5. Latency of n-th arriving unique response - WDM, 300 content items.

Figure 5 shows the latency from a query to arrival of the n-th unique response in the WDM scenario. In cases where the first response drop already satisfies the search, the content items are retrieved from the vicinity of the querying node. This results in low latencies for the search when first contact routing is used. Spray-and-wait leads to relatively similar latency profile. The latency is always highest with epidemic routing, but the approach yields a good success ratio as discussed above.

V. Implementation Considerations

We have created an experimental implementation of a search capable node following the outline of Figure 6. This separates the bundle layer functionality into a Bundle Protocol Agent, search specific functionality into a DTN-Search client, and application specific functionality into a separate application running on top of DTN-Search.

The Bundle Protocol Agent sends and receives bundles over the network, makes routing decisions, and provides persistent storage to facilitate *store-carry-and-forward* operation. Applications can access the BPA functionality through the *Client API* to send and receive bundles. This API is also used by DTN-Search to handle incoming and outgoing queries and responses. The basic API is capable of supporting search mechanisms that do not require interactions beyond forwarding bundles. This is enough to implement TTL and hop-count based search termination mechanisms which can be implemented fully in the bundle layer. However, first response filtering and global response count estimation based termination mechanisms require the ability for the application to control the forwarding on the query by: 1) signaling to the router that the bundle should be dropped form the queue (terminating the query), and 2) modifying the contents of the bundle before it is forwarded (attaching the number of

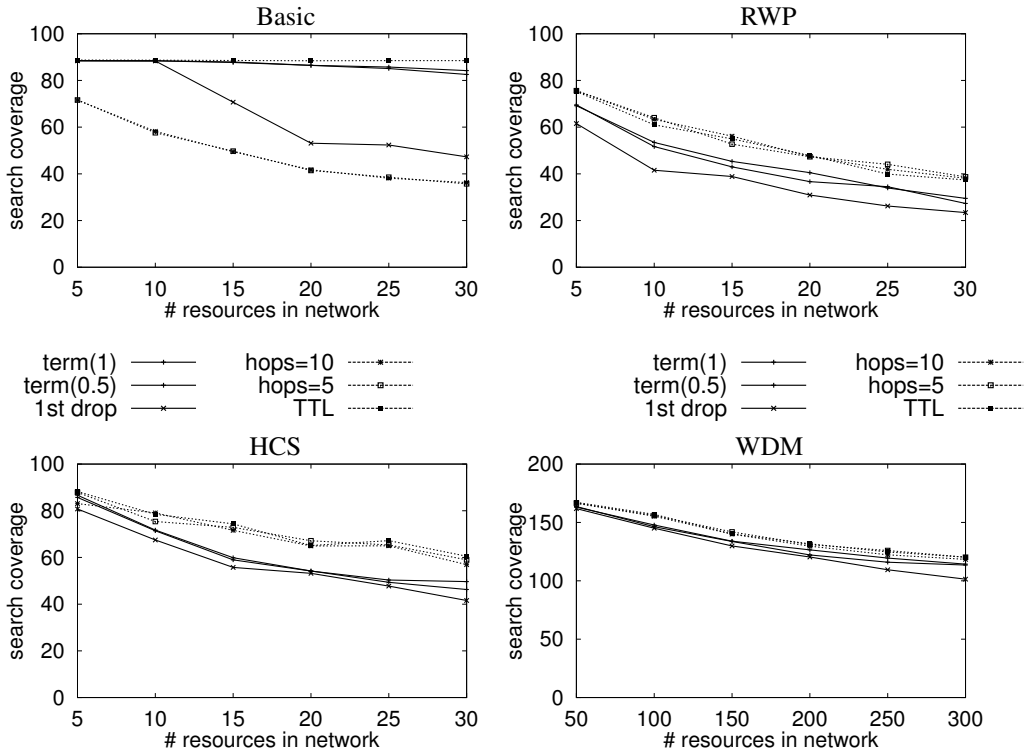

Fig. 4. Search coverage with Epidemic routing.

responses generated).[4]

Search processing takes care of evaluating queries, response generation, and search termination as described in section III. The search layer has its own storage for content items optimized for content retrieval.

The query evaluation process could be done directly on the contents of the items in the application storage by, e.g., doing string matching. However, in the general case this approach has a number of drawbacks: 1) the cost of matching operations is likely to become significant when the size of the application storage grows, and 2) the query logic must understand the structure of the content items (e.g., in order to search based on ID3 fields of an MP3 file the query logic must be able to parse the ID3 tags from the content items). Instead we suggest a generic, metadata based mechanism for query evaluation: Each content item can have an arbitrary number of attached metadata key-value pairs with well known value types (e.g., strings or numeric values) stored in a structured index. The metadata needs to be generated only once when the content item is inserted into the network, but later generation of additional metadata by other nodes is also possible. Queries can then be composed of a combination of logical operations and comparisons of the metadata values. These generic queries can then be evaluated by any DTN-Search node while the precise semantics of the metadata fields are only known at the application layer.

[4]These two conditions are not met by the DTN2 reference implementation and its external router interface.

DTN-Search exposes a simple, query-response *Search API* that can be used by applications to place search queries and to asynchronously receive responses as they become available. The API can also be used by the applications to insert new content items and associated metadata into the search layer.

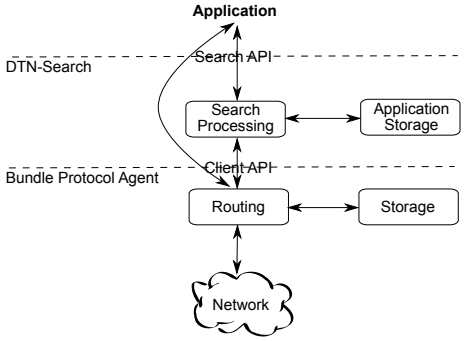

Fig. 6. Components of the DTN-Search Implementation

This separation of concerns between the bundle router and the search logic has several advantages: As the bundle router itself is not modified, it is fully compatible with nodes that do not implement search. The search storage can be optimized for content retrieval while router's storage is optimized for access based on routing information such as source, destination, and lifetime. Furthermore, the routing mechanism can be adapted to the needs of specific scenarios instead of being fixed on a generic search routing.

There are also a few disadvantages of this approach compared to a modified bundle router for DTN-Search: The redundant storage may be problematic for resource-constrained devices. Also, the search processing cannot make routing decisions which can lead to a suboptimal performance.

VI. Conclusions

In this paper, we have investigated searching for contents in mobile DTNs (without internal structure or other content organization), focusing on controlling the spreading of search requests in an otherwise unloaded network. Looking at three different routing protocols and four different mobility scenarios, we have analyzed two network and two application level mechanisms (with different parameterizations). As expected with DTNs, content items are often retrieved from the vicinity and particularly widespread resources are likely to be found close by. In the scenarios we investigated, node mobility and the DTN-inherent message delivery delays make attempts for exhaustive searches unattractive: popular contents will likely be found, items from the long tail only by chance. This suggests using content dissemination or exploiting geographic proximity in conjunction with searching in mobile DTNs.

While the investigations in this paper provide a starting point in analyzing various properties of a basic search system in mobile DTNs, numerous directions become apparent which deserve further attention: We currently consider only statically distributed resources; however, a node which successfully retrieved a resource will subsequently also hold a copy and thus may respond to future queries, too. Similarly, nodes temporarily holding copies of resources (identified via metadata) while forwarding them could copy the reply to third parties seeking the same information, thus effectively acting as caches [17]. Such application-routing interaction could also be leveraged to suppress duplicate replies from different responders as mentioned earlier. And nodes could continuously overhear response messages as they forward them to obtain an estimate of the resource distribution and then use information from earlier queries and the current one as additional input for the forwarding/termination and selection phases.

Acknowledgements

This work was partly funded by the Academy of Finland in the DISTANCE project (grant no. 117429) and by Teknologiateollisuus ry in the REDI project.

References

[1] L. A. Adamic, B. A. Huberman, R. M. Lukose, and A. R. Puniyani. Search in power law networks. In *Physical Review E*, volume 64, October 2001.

[2] Chen Avin and Carlos Brito. Efficient and robust query processing in dynamic environments using random walk techniques. In *IPSN '04: Proceedings of the third international symposium on Information processing in sensor networks*, pages 277–286, 2004.

[3] Aruna Balasubramanian, Yun Zhou, W. Bruce Croft, Brian Neil Levine, and Aruna Venkataramani. Web search from a bus. In *CHANTS '07: Proceedings of the second workshop on Challenged networks CHANTS*, pages 59–66, 2007.

[4] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *INFOCOM*, pages 126–134, 1999.

[5] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H.Weiss. Delay-Tolerant Network Architecture. RFC4838, 2007.

[6] Nicholas B. Chang and Mingyan Liu. Controlled flooding search in a large network. *IEEE/ACM Trans. Netw.*, 15(2):436–449, 2007.

[7] Karen Church, Barry Smyth, Paul Cotter, and Keith Bradley. Mobile information access: A study of emerging search behavior on the mobile internet. *ACM Trans. Web*, 1(1):4, 2007.

[8] Frans Ekman, Ari Keranen, Jouni Karvo, and Jörg Ott. Working day movement model. In *Proceedings of the 1st SIGMOBILE Workshop on Mobility Models for Networking Research*, May 2008.

[9] Pan Hui, Jeremie Leguay, Jon Crowcroft, James Scott, Timur Friedman, and Vania Conan. Osmosis in Pocket Switched Networks. In *ChinaCom*, 2006.

[10] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in Delay Tolerant Networks. Proceedings of the ACM SIGCOMM 2004 Conference, Portland, OR, USA, 2004.

[11] Bernard J. Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998.

[12] Bernard J. Jansen, Amanda Spink, and Jan Pedersen. A temporal comparison of altavista web searching: Research articles. *J. Am. Soc. Inf. Sci. Technol.*, 56(6):559–570, 2005.

[13] Maryam Kamvar and Shumeet Baluja. A large scale study of wireless search behavior: Google mobile search. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 701–709, 2006.

[14] Gunnar Karlsson, Vincent Lenders, and Martin May. Delay-tolerant broadcasting. In *Proceedings of ACM SIGCOMM Workshop on Challenged Networks (CHANTS)*, pages 197–204, September 2006.

[15] A. Keränen and J. Ott. Increasing Reality for DTN Protocol Simulations. Technical report, Helsinki University of Technology, Networking Laboratory, 2007.

[16] Jeremie Leguay, Anders Lindgren, James Scott, Timur Friedman, and Jon Crowcroft. Opportunistic content distribution in an urban setting. In *CHANTS '06: Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, pages 205–212, 2006.

[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 and Jörg Ott. Redundancy and Distributed Caching in Mobile DTNs. In *Proceedings of the 2nd ACM MobiArch Workshop*, August 2007.

[19] Keith L. Scott and Scott C. Burleigh. Bundle Protocol Specification. RFC5050, November 2007.

[20] Sergio D. Servetto and Guillermo Barrenechea. Constrained random walks on random graphs: routing algorithms for large scale wireless sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 12–21, 2002.

[21] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[22] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *WDTN '05: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pages 252–259, 2005.

[23] Susan Symington. Delay-Tolerant Networking Metadata Extension Block. Internet Draft draft-irtf-dtnrg-bundle-metadata-block-00, Work in progress, September 2008.

[24] Wesley W. Terpstra, Jussi Kangasharju, Christof Leng, and Alejandro P. Buchmann. Bubblestorm: resilient, probabilistic, and exhaustive peer-to-peer search. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 49–60, 2007.

[25] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University, April 2000.

[26] Bo Yang and Ali R. Hurson. Content-aware search of multimedia data in ad hoc networks. In *MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 103–110, 2005.