Web Search From a Bus

Aruna Balasubramanian   Yun Zhou
W. Bruce Croft   Brian Neil Levine   Arun Venkataramani
Dept. of Computer Science, University of Massachusetts Amherst
{arunab, yzhou, croft, brian, arun}@cs.umass.edu

ABSTRACT

Opportunistic connections to the Internet from open wireless access points is now commonly possible in urban areas. Vehicular networks can opportunistically connect to the Internet for several seconds via open access points. In this paper, we adapt the interactive process of web search and retrieval to vehicular networks with intermittent Internet access. Our system, called *Thedu*, has mobile nodes use an Internet proxy to collect search engine results and prefetch result pages. The mobile nodes download the pre-fetched web pages from the proxy. Our contribution is a novel set of techniques to make aggressive but selective prefetching practical, resulting in a significantly greater number of relevant web results returned to mobile users. In particular, we prioritize responses in the order of the usefulness of the response to the query, which allows the mobile node to download the most useful response first. To evaluate our scheme, we deployed Thedu on Diesel-Net, our vehicular testbed operating in a micro-urban area around Amherst, MA. Using a simulated workload, we find that users can expect four times as many useful responses to web search queries compared to not using Thedu's mechanisms. Moreover, the mean latency in receiving the first relevant response for a query is 2.7 minutes for our deployment; we expect Thedu to have even better performance in larger cities that have densely populated open APs.

Categories and Subject Descriptors

C.2 [**Computer Communication Network**]: Network Architecture and Design—*Wireless communication; Store and forward networks*

General Terms

Design, Performance

Keywords

Disruption-tolerant networking, web search, application, testbed

1. INTRODUCTION

Internet connectivity is commonly available from open wireless access points (APs) in many urban areas. Typically, such APs use 802.11 and are deployed for stationary users. However, the range of 802.11 also allows for connectivity for seconds or more at a time from moving vehicles, as several testbeds have demonstrated both in urban [3, 6, 12] and developing regions [14, 16, 9]. The density of APs and the speed of the vehicles largely determine the duration of connection and disconnection in such vehicular networks.

Few networked applications can be deployed *as is* for vehicular users when only intermittent connectivity is available. For example, web browsing is adaptable in such intermittent networks if the user requests a specific URL; however, web sessions more commonly begin with queries to a search engine, including Yahoo, Google, or MSN. The search engines return numerous links to web pages, some of which are then retrieved by the user. The process ends when the user has found and downloaded one or more web pages that are *relevant* to the query. Adapting this interactive process is not viable given that periods of connectivity can occur unexpectedly and be shorter than user think times.

In this paper, we adapt the commonly interactive process of web search and retrieval to vehicular testbeds that see only opportunistic connectivity. In our system, called *Thedu*[1] an Internet proxy collects search engine responses for queries issued by a mobile node and pre-fetches the web pages corresponding to each response. The mobile user then downloads the pre-fetched pages from the proxy when Internet connectivity is available. Downloading all pre-fetched pages is inefficient and wasteful of the limited bandwidth available. Our contribution is a novel set of techniques to prioritize pre-fetched web pages, so that a mobile user downloads a significantly greater number of relevant web pages and avoids wasting bandwidth on responses that are unlikely to be useful. First, using a simple classifier, the proxy identifies queries that likely require only a single response (known as *homepage queries* [2]). Second, for other queries the proxy prioritizes web pages according to the estimated relevance of the page to the original query. Notably, we contribute a new technique that normalizes the relevance scores of pages across disparate queries, which allows the proxy to prioritize web pages for different queries requested by the user.

To evaluate our scheme, we deployed Thedu on DieselNet, our vehicular testbed of 40 buses operating in a 150 sq. mile, micro-urban area around Amherst, MA. Using a simulated

---

[1]Thedu is the Tamil word for *search*. The first syllable rhymes with "hay" .
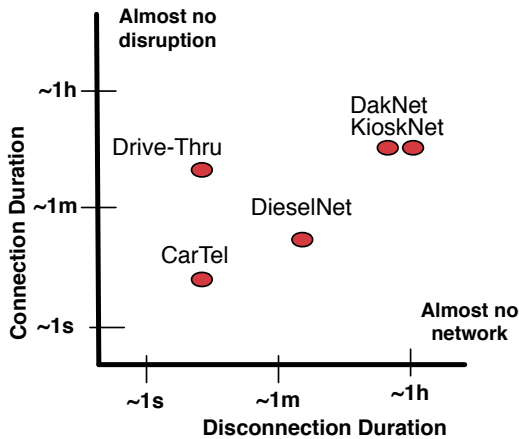
Figure 1: DTN testbed connectivity characterstic

workload we find that the mean latency in receiving the first relevant web page for a query is 2.7 minutes. We also find that users can expect four times as many useful web pages retrieved in response to queries compared to when not using Thedu. We believe that providing a web search and retrieval service in vehicular disruption-tolerant network (DTNs) increases the utility of such deployments. In the following sections, we detail our model and assumptions, describe the challenges of supporting web search in disconnected environments, and detail our solution and deployment.

2. RELATED WORK

2.1 Disruption-Tolerant Networks

Several DTN testbeds have been deployed to provide Internet connectivity to moving vehicles via open access points. Figure 1 quantitatively compares four DTN testbeds by the reported observed median frequency of disruption and connectivity. DakNet [14] and KioskNet [16] transfer data from cities with access to Internet to smaller villages. The disconnection between the vehicles and APs in the city is on the order of hours. Our testbed, DieselNet [3], is deployed in a mixed rural and urban region, and we observe that the inter-meeting time between a bus and an access point is in the order of several minutes. CarTel [6] and the Drive-Thru Internet platform [12] are set in urban environments and connect to APs more frequently. The CarTel [6] study reports that disconnections between a mobile node and any open Internet access point is typically about 75 seconds. In most areas except small urban pockets, vehicular connectivity is likely to be intermittent and not always predictable [21].

Ott et al. [13] proposed an architecture for HTTP applications over vehicular DTNs to retrieve content from a known URL. The web search application we seek to provide with Thedu is less precise than an URL-based, document retrieval application. In web search, several responses are retrieved for one query, and one or more of them may be relevant for the query. In document search, on the other hand, precisely one document is retrieved and its location is known in advance. To deploy an efficient web search application, we integrate novel IR techniques in our solution that increase the percentage of relevant documents returned to the user. We determine the usefulness of each response returned for a

query and prioritize responses in the order of their usefulness. We use a technique similar to that proposed by Ott et al. [13] of returning bundled responses to users.

The proxy scenario considered in Thedu is similar to the model used in the *infostation* project [4], where base stations store information for mobile nodes and a mobile node downloads the information when connected. There have been many applications developed using the infostation model, including more recently, supporting distribution of movies [20]; however, past work has not examined the provision of web search at the infostation.

Currently Thedu uses only open APs to route application specific information to mobile nodes. Several recent DTN work focus on using the mobile nodes themselves to route data among each other [7, 3, 18]. In future work, we hope to use the bandwidth available among peers from DTN routing to improve application throughput.

2.2 Information Retrieval

Recent related works in IR for mobile devices have not addressed environments with frequent disconnection. This excludes our own past work [5], which assumed that the database is partially replicated and carried by the mobile peers themselves, rather than the database being accessible only via the Internet.

Our main IR contributions is an extension of recent work on *language models*. Language models provide a framework for retrieving documents for a query from a document collection. For example, in *query-likelihood* [17] models, each document in a single collection is given a score according to the likelihood that the query was generated for a given document. The documents are returned in the decreasing order of their score. The document scores are normally not comparable across different queries or different document collections. For example, Lu and Callan [10] provide a language modeling framework for normalizing document scores from different collections. In contrast, the Thedu proxy normalizes scores across different queries. More specifically, our scoring technique answers the following question: Is document A for query Q_1 more likely to be relevant than document B for query Q_2?

3. WEB SEARCH ON DTNS

Our goal in designing Thedu is to mask moderate disruption faced by users and provide a web search interface that allows users to search for relevant web pages, albeit with a higher latency. When the disconnection periods are in the order of hours, the service will satisfy few participants. However, we leave the significant task of measuring the exact tolerance of users to delayed delivery for our later work. We assume a DTN comprised of vehicles (or pedestrians) that have intermittent connectivity to the Internet through open wireless APs. Using the APs, mobile nodes have temporary access to any resource on the Internet. We also assume that each query is assigned a deadline for delivery. In this section, we review the challenges that DTNs pose for web search specifically, and we detail Thedu's design.

3.1 Web Search Challenges

Deploying an efficient web search application on disruption-prone vehicular networks *as is* is not possible for several reasons. A web search application begins with a query issued by a user on a mobile DTN node. Ideally, when the node gains
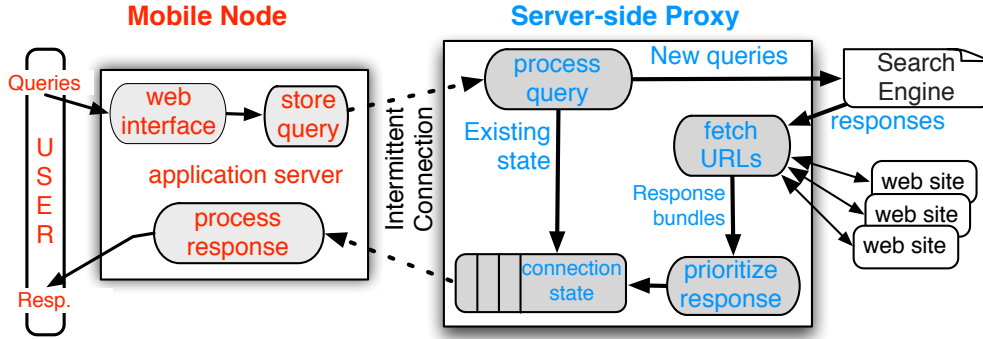
Figure 2: Thedu Architecture

Internet access, the search application immediately connects to a search engine, sends the query, and receives responses. The search engine returns responses for a query in the form of URLs and document previews (i.e., *snippets*). The user then retrieves the web-pages associated with one or more URLs before the connection ends. Users typically click and retrieve one web-page at a time. This serial processing and think time will work against finding and retrieving responses before the connection is lost. If disconnection is in the order of minutes or hours, this interactive process is untenable.

When the exchange is disrupted, it may be that only a subset of responses can be returned to the mobile node. Search engines do not maintain state information regarding incomplete downloads. When the node regains access, the process must begin anew: the query needs to be reissued and the response retrieved from the search engine. The performance worsens when multiple users on a bus compete for bandwidth during periods of connection.

3.2 A Proxy-based Solution

Figure 2 shows our Thedu architecture. The mobile node accepts user query terms through a web interface. The queries are stored until a connection is found to a proxy through an open AP. The queries are sent with a nonce that is used to re-identify the request if the connection is broken. The mobile node downloads responses in the form of web pages from the proxy. If the connection is broken, the remaining web pages are downloaded at the next opportunity.

At the server side, a proxy awaits connection from a mobile node. If the mobile node has pending responses, it downloads the responses immediately after connecting to the proxy. In parallel, the proxy sends new query requests to the search engine. The search engine returns URLs and document snippets in the order of their relevance to the query. The proxy pre-fetches the web pages associated with the URLs in parallel. The proxy also retrieves all images and other objects and places them in a bundle. Jansen et al. [8] have shown that 75% of users do not view more than the top 20 results and most users are interested in at most 5 results per query. Therefore, the Thedu proxy needs only to pre-fetch top documents for a high probability of satisfying the user.

The proxy prioritizes response bundles as it places them in a node-specific outgoing queue. The bundles are prioritized according to relevance and query-type; we describe the prioritization technique in detail in Section 4. Although the queue is re-prioritized with each new bundle, on-going downloads are not preempted. Finally, we note that when the proxy receives multiple queries from one mobile node, it retrieves responses for each query in parallel.

4. WEB PAGE PRIORITIZATION

Thedu's prioritized list of web pages is in the decreasing order of their utility to the end-user. We note that all search engines provide a ranked order of the web pages. We can use the same ranking for prioritization. Although simple, this strategy is not suitable because the rank cannot be used to prioritize web pages retrieved for different queries. Some search engines provide a more specific *relevance score* for each response to quantify the relevance of the web page to the query. However these scores are also not comparable across queries.

Thedu prioritizes web pages using two key ideas. First, it uses a *normalization* method so that relevance scores of web pages can be compared across queries. Second, it classifies the *query type* to determine user intent and how many relevant web pages the user desires. We detail each mechanism in this section.

4.1 Query normalization

We explain the query normalization technique as a modification of the Indri [19] search engine. Indri [19] is an academic search engine shown to be effective for web search. Indri returns a ranked set of documents in response to a query and associates each document with a relevance score. In case of web search, a document is a web page.

In the Indri model, the *relevance score* of a document D for query Q is estimated as

$$Score(D) = \prod_{w \in Q} \lambda P(w|D) + (1 - \lambda)P(w|C) \qquad (1)$$

where λ is a smoothing constant, $P(w|D)$ is the probability of a word w occurring in a document D and $P(w|C)$ is the probability of the word w occurring in C, the entire collection of documents being searched.

Since the document score depends on the words in the query and the collection, the scores of documents for different queries (or over different collections) are not comparable. To normalize document scores across queries we propose a score normalization method that computes scores as:

$$Score(D) = \frac{1}{|Q|} \sum_{w \in Q} log \frac{\lambda P(w|D) + (1 - \lambda)P(w|C)}{P(w|C)} \qquad (2)$$

Homepage	Content
The query terms or acronym occur in the URL	The query is in form of a question
All query terms occur in the title and anchor text	One of the top three URLs is a "wiki" (commercial)
The query is less than 3 words	The query is greater than 3 words
The URL is a root	

Table 1: Features used to classify homepage and content queries

	Homepage	Content
Training size	45 queries	25 queries
Testing size	100 queries	25 queries
Accuracy	88%	73%

Table 2: Classification results from Indri

Note that the document rank remains unchanged by normalization, as does the search effectiveness. In our deployment, we set $\lambda = 0.4$. A full derivation and explanation of Eq. 2 is available in a technical report [22].

4.2 Query-type classification

Understanding the user's intent can help increase the usefulness of responses and limit useless prefetching. For example, returning only one web page suffices for a user's query of "Chants 2007", even if other web pages have a high relevance score.

Broder [2] classifies web queries into three types: *homepage*, *content*, and *service queries*. *Homepage* queries try to find known items; e.g., "computer science umass". *Content* queries seek an answer or a meaning and not a specific website; e.g., "kosovo conflict". Such queries may be ambiguous or opinion-based with the user seeking several relevant web pages. *Service* queries are location-specific queries such as "sushi bar amherst".

Thedu classifies queries as either homepage or non-homepage (i.e., content or service) and determines the number of responses to return accordingly. We use a naive bayes classifier to classify queries into query types. We enumerate characteristic features of homepage and non-homepage queries and use these features to train the classifier. The features are summarized in Table 1. The features only depend on the URL, snippet, and title fields. We trained the classifier on a set of queries using the URL, titles, and *snippets* fields for the top 10 web pages returned for each training query.

For example, for the training query "prime factors", the URL, title and *snippet* for the first response from the Google search engine is

- <url> http://www.gomath.com/algebra/factor.php </url>
- <snippet> To prime factor a number, begin dividing by the smallest possible prime and continue until the quotient is a prime number. </snippet>
- <title> prime factor </title>

We note that the URL, title, and snippet fields are short and easy to parse. Our technique is also independent of characteristics of the collection of documents. Therefore our classifier allow us to classify query-type without assistance from the search engine and as queries arrive.

Classifier Evaluation. We analyzed the performance of the classifier on Google search engine and Indri [19] search engine. We trained the classifier using the queries from TREC. NIST Text Retrieval Conference (TREC) is a research standard for evaluating Information Retrieval performance. TREC provides collections of documents and queries that are pertinent to the documents in the collection. We used a

set of TREC queries to train the classifier and tested on a different set of TREC queries from 2001 TREC web-track.

When using Google, we retrieved the top 10 responses, URLs, and snippets for the training queries using the Google API (code.google.com). When using Indri, we queried the Indri search engine to retrieve the top 10 responses for the queries. To use Indri, we indexed W-10G web document collection. Indri retrieved responses for the queries from this collection because the queries in the 2001 TREC web-track are pertinent to the W-10G document collection.

The TREC database already separates queries into homepage and non-homepage queries. We evaluate the performance of our classifier by comparing against the corresponding TREC categorization. Table 2 presents the results of our classifier on Indri. When tested on Google, the accuracy of homepage prediction was 90% and the accuracy of non-homepage prediction was 71%. We note that using a simple set of features and training on a small set of queries and responses, our classifier is able to predict query type accurately.

4.3 Prioritization Algorithm

We now describe Thedu's prioritization algorithm, which combines the normalization, classification, and probability results above.

First, we calculate the probability that a document is relevant for a query. Since most search engines only return relevant scores or a ranked list, we calculate the relevance probability from the document score. We use a technique proposed by Manmatha et al. [11] that models documents scores of a relevant document, $P(score|rel)$, as a gaussian distribution and scores for non-relevant documents, $P(score|nonrel)$, as an exponential distribution for a language-model based search engine (e.g., Indri). Accordingly, Thedu estimates $P(r)$, the probability that a document r is relevant given the score as

$$P(r) = P(rel|score) = \frac{P(score|rel)P(rel)}{P(score|rel)P(rel) + P(score|nonrel)P(nonrel)} \quad (3)$$

We estimated the relevance probability of a document $P(r)$ using Eq. 3 by estimating the parameters for the Gaussian distribution ($P(score|rel)$) and Exponential distribution ($P(score|nonrel)$). We used a training data set of 25 content queries and 45 name-page finding queries from 2001 TREC web-track. We retrieved responses for the training data, classified them as relevant and non-relevant and used their relevance scores to determine the parameters of the gaussian and exponential distribution as $N(\mu = 0.669, \sigma = 0.00000517)$ and $exp(\lambda = 1.489)$.

Thedu works on web collections where documents are web pages. Let $E(q)$ be the estimated number of relevant web pages returned for query q. $E(q)$ is calculated as $\sum_{r=1}^{n} P(r)$ (see Robertson [15]), where n is the number of returned web pages.

The Thedu prioritization algorithm for web pages is as follows:

1. Rank web pages using normalized scores.
2. If query has expired or is of type homepage and user sent a feedback that an earlier web page response to the query was relevant:
   (a) Remove query and all associated web pages.
3. Else
   (a) Get the next web page r in the sorted list. Let the query corresponding to the web page be q.
   (b) Update $E(q) = E(q) + P(r)$.
   (c) If query is is classified as homepage, and if $E(q) > 1$; i.e., the expected number of relevant web pages sent for the homepage query is greater than 1: skip the response.
   (d) Else, return the web page

The prioritization algorithm returns responses in the decreasing order of their score. But when the query type is a home-page, only one relevant response need to be returned. In a single transfer opportunity, Thedu returns web pages for the query until at least one of them is likely to be relevant (in expectation).

5. DEPLOYMENT

We implemented and deployed Thedu on DieselNet [3], a vehicular DTN testbed consisting of 40 public transit buses. Each of the 40 buses in the DieselNet testbed is equipped with a small-form Linux computer (40 GB of storage and 256 MB of RAM). The buses operate an 802.11b radio that scans for open Internet APs. For a fuller discussion of Diesel-Net see Burgess et al. [3]. Here we characterize bus-to-AP interactions, which are not covered in Burgess et al.

The buses each carry an implementation of the client-side proxy, shown in Figure 2, and connect to the server-size proxy. We deployed two version of the server-side proxy: Thedu proxy and a *stateless* proxy. As we describe below, the stateless proxy is equivalent to using no proxy. We deployed the Thedu proxy from March 26–30, 2007 and the stateless proxy from May 7–11, 2007[2].

In sum, we found that the Thedu proxy returns 4 times as many relevant web pages as the stateless proxy and returns at least one relevant web page per query, which is twice as many as the stateless proxy.

5.1 Experimental Setup

We desired to use Yahoo, Google, or similar commercial search engines in our experiments, however none provide relevance scores; instead they produce only ranked lists of results. Therefore, for both deployments, we used the Indri [19] search engine. In our future work, we are exploring techniques that will allow us to use a commercial search engine for our deployment.

The two proxy experiments differ as follows.

- **The Thedu proxy** is implemented as described in Section 3.2.
- **The stateless proxy** strips features from the Thedu proxy so that it is equivalent to the case where mobile nodes do not use a proxy. The stateless proxy retrieves web pages for queries using Indri and returns bundles to the client in FIFO order (i.e., by completed retrieval

---

[2]Traces are available at traces.cs.umass.edu.

Collection:	WT10g [1]
Number of queries:	150
Homepage queries:	TREC 2001 Homepage-Finding Topics
Content queries:	TREC 2001 Ad hoc Topics
Search engine:	Indri
Query deadline:	30 min
Queries per hour:	10 per bus

Table 3: IR parameters used for deployment.

time). The stateless proxy terminates all incomplete transactions when the client disconnects. When the client reconnects, retrieval begins anew.

For the Thedu proxy, we modified the Indri source code to provide normalized scores as described in Section 4. In both proxies, we removed queries and associated web pages after 30 minutes (since we assume most bus passengers would exit the bus by that point). The deployment parameters are tabulated in Table 3.

We used Indri to index and stored a standard collection of web pages, from TREC WT10G web collection [1]. We used standard queries from 2001 TREC web-track associated with the WT10G collection for evaluation.

TREC also provides an "answer key" called the relevance judgment. The relevance judgments lists all documents that are relevant to a query as determined by human subjects. Of course, the proxy does not have access to the relevance judgement when returning web pages but we use the judgments for evaluation. We note that for homepage queries, relevance judgment normally lists only one document as being relevant.

We pre-loaded the buses with 150 queries from web-track 2001. Unfortunately, the queries associated with the WT10G collection are limited and relevance judgments are only available for the TREC prescribed queries; so we are limited to 150 queries. This query set and evaluation technique is commonly used in the Information retrieval community to measure retrieval effectiveness. The client in the bus periodically generated a *(queryID, query)* pair, where queryID is a monotonically increases sequence number and the query is chosen from the pre-loaded queries. The purpose of the queryID is to allow a query to be repeated as if it were completely new — neither the client nor proxy use a cache of previously downloaded content for this reason, but a deployment for real users would employ a cache to exploit query and response locality.

For all experiments, queries are generated with an average of 10 per hour per bus with inter-arrival times drawn from an exponential distribution. At each connection to a AP, the bus connects to the proxy and sends all queries generated since the last connection. The buses periodically upload statistics of queries, responses, and delays. Delays are calculated against each query's generation time stamp; that is, delays include time accumulated while waiting for the next open AP.

Finally, we note that we perform experiments with the two proxies on two different weeks. Simultaneous real deployment of both systems is not possible; moreover, each bus should be viewed as a separate experiment of the system. Some routes have more APs than others and so partial deployment of each on the same day would skew the experiment. Since we cannot control which scheduled route each bus serves each day, it is more accurate to deploy each system separately.
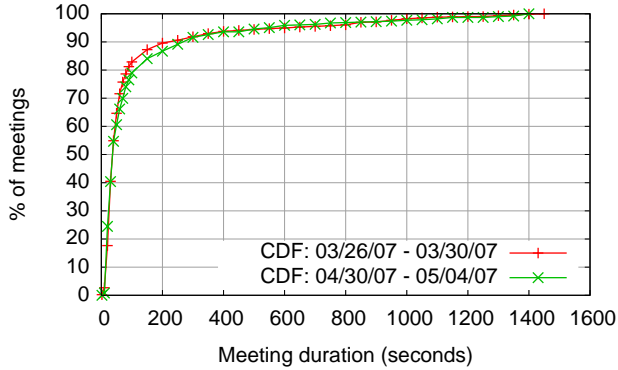
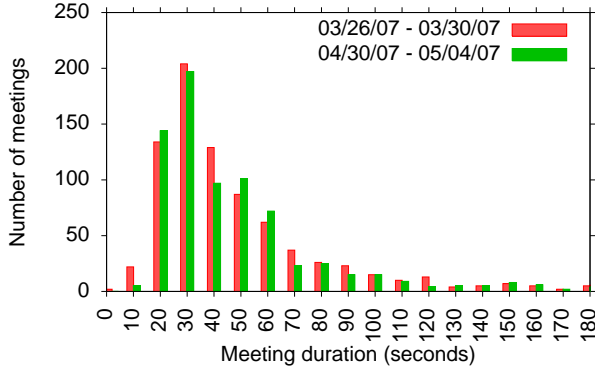Figure 3: CDF of Bus-to-AP meeting durations. Median of 45 seconds for both sets.

Figure 4: Bus-to-AP interactions less than 3 minutes.

5.2 Network characteristics

Figure 3 shows the duration bus-AP meetings during the two separate measurements; the two weeks show similar results. In both cases, about 80% of all meetings last for less than 50 seconds. Less than 5% of meetings last more than 400 seconds. These outliers result from buses that have powered-on but idle engines when in the garage (which has an AP) or buses that wait in traffic or at a bus stop while next to an AP. For both weeks, each set had hundreds of meetings with APs that lasted 10–180 seconds, as shown in Fig 4.

Figure 5 shows the bus-to-AP inter-meeting time. We ignore disconnections if the bus returns to the garage in the middle of the day. Here we see some dissimilarity between the two data sets. In the week Thedu proxy is deployed, the number of shorter disconnections are marginally higher compared to the week when the *stateless* proxy is deployed. In other words, the mean inter-meeting time is 5 minutes in the earlier week and 8 minutes in the latter week. The shorter delay between meetings for the early data set lowers latency for receiving results for Thedu; however, as we show in this section and the next, the difference in performance between Thedu and stateless proxy is large and cannot be discounted based on this small dissimilarity in connection characteristics.

5.3 Thedu Performance

The results of our experimental deployments are presented in Table 5. Each result is a per-day average.
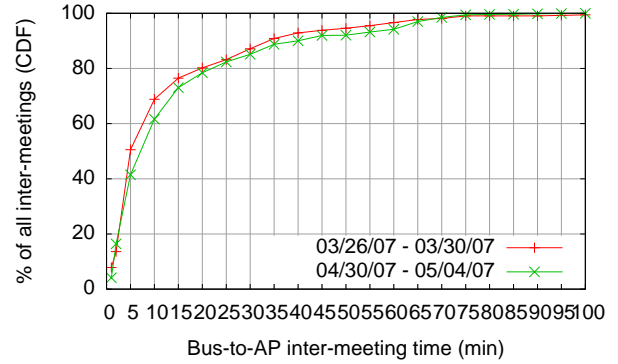

Figure 5: CDF of bus-to-AP inter-meeting times. Median of 5 and 8 minutes for earlier and later set, respectively.
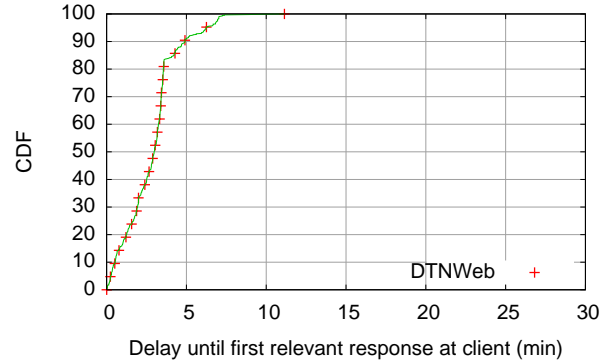

Figure 6: Average (per day) delay in getting relevant web pages

Statistic	3/26–3/30	4/30–5/04
Number of meetings	897	935
Avg meeting duration (sec)	54.75	57.59
Avg bandwidth(Kbps)	59.3	57.6

Table 4: Average per day statistics

We note that application development on DTNs requires that the performance be measured using an application-specific metric. For example, the number of relevant web pages sent by the Thedu proxy is a more direct measure of the application performance than the number of web pages returned. The performance of Thedu is better across a number of application metrics. First, Thedu was able to completely use the bandwidth available at each connection opportunity to return web pages, while the stateless proxy suffered from extra processing during each connection. Second, Thedu was able to multitask by sending pending web pages while retrieving results from new queries. Third, Thedu prioritized responses to send the most useful web page first.

Accordingly, the Thedu was able to return more than four and half times as many web pages per query on average. More importantly, by prioritizing useful responses, the number of relevant web pages sent by Thedu is four times larger than the stateless proxy. Finally, Thedu was able to send at least one relevant web page to twice as many queries compared to the stateless proxy.

Statistic	Thedu Proxy	Stateless Proxy
Queries:	780	743
Web pages:	5639	1207
Avg resps. per query:	7.2	1.6
Relevant web pages:	1630	401
Queries with at least 1 relevant web page:	529 (68%)	291 (39%)

Table 5: Average per day statistics

Finally, we measure the delay of receiving a relevant web page for a query. Figure 6 shows an empirical CDF of the delay before receiving the first relevant web page at the client. We note that 90% of the time, the first relevant web page is received within 5 minutes and the mean delay is 2.7 minutes. These delay measurements are encouraging — they show that a DTN web search application is feasible even in semi-urban settings that has a deployment of access points that is sparser than what is found in larger cities.

5.4 Trace-Driven Simulations of Load

To simulate the effect of higher query load, we ran trace driven simulations of our system. Our traces of bus-to-AP transfers in DieselNet record the time when a bus met an access point and the duration of meeting.

We used the trace-driven simulation experiments to compare the performance of Thedu to two variants: *(i) Thedu without the query-type classification* that we describe in Section 4.2; and *(ii) Round-Robin* allocation of bandwidth for downloading responses. Last, we compare the performance of Thedu with a *stateless* proxy similar to our deployment.

Thedu without the query-type classification prioritizes responses based on relevance probability without considering the type of query (i.e., topic or homepage). The *Round-Robin* variation uses round-robin prioritization and allocates equal bandwidth to all queries. *Round-Robin* neither uses query-type or relevance.

Figure 7 shows the total number of web pages delivered as query load increases. The results are an average of 10 trials per point on the graph. The vertical lines at each point in the graph show the 95% confidence interval. The experiment shows that Thedu increases the number of relevant web pages by over 28% compared to a round-robin scheme. The experiments also show that the improvement in performance when using Thedu over *Round Robin* is statistically significant for loads greater than 5 queries per hour. Interestingly, we are only able to achieve this improvement when we predict the query-type with our classification algorithm; Thedu performs about 12% worse when it does not classify query-type. This is because, when Thedu does not use query-type classification, it returns several responses for a homepage query when only one is relevant, wasting bandwidth.

Our results also indicate that maintaining state provides the most benefit in terms of performance. *Stateless* proxy performs about 40% worse than *Round Robin* and about 70% worse than Thedu.

6. CONCLUSIONS

We are interested in unifying architectural themes across different kinds of edge networks, including MANETS, mesh networks, and delay-tolerant networks. We believe that a principled understanding of this broad question must include
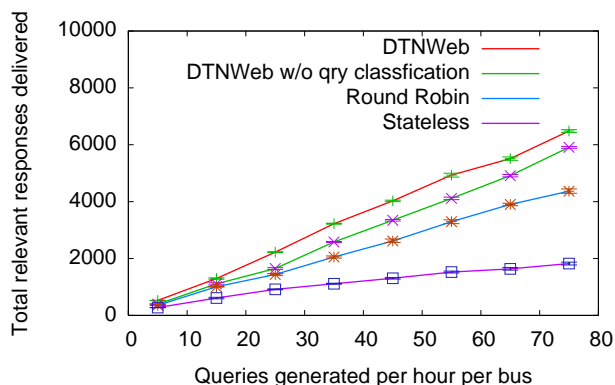

Figure 7: Number of relevant web pages delivered

developing real applications over challenged networks. Our case study of web search over DTNs is a first step in this direction.

Our preliminary results are encouraging. Based on an evaluation over a vehicular testbed, we find that users can expect useful responses to web search queries in an average of 2.7 minutes in a semi-urban city with sparsely deployed access points. Our system, Thedu, introduces two novel ideas — one-shot queries and and relevance normalization — and synthesizes them with existing techniques into an architecture that we believe holds useful lessons for building more DTN applications.

Acknowledgments

7. REFERENCES

[1] Text retrieval conference (trec).
    http://trec.nist.gov/.
[2] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.
[3] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *Proc. IEEE INFOCOM*, April 2006.
[4] D. J. Goodman, J. Borras, N. B. Mandayam, and R. D. Yates. INFOSTATIONS: a new system for data and messaging services. In *Proc. IEEE VTC*, pages 969–973, 1997.
[5] K. M. Hanna, B. N. Levine, and R. Manmatha. Mobile Distributed Information Retrieval For Highly Partitioned Networks. In *Proc. IEEE Intl. Conference on Network Protocols (ICNP)*, pages 38–47, Nov 2003.
[6] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *Proc. ACM SenSys*, pages 125–138, 2006.
[7] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *ACM SIGCOMM*, pages 145–158, 2004.

[8] B. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Information Processing and Management*, 36(2):207–227, 2000.

[9] A. Lindgren, A. Doria, and O. Schelén. Probabilistic routing in intermittently connected networks. In *Proc. SAPIR Wrkshp*, 2004.

[10] J. Lu and J. Callan. Content-based retrieval in hybrid peer-to-peer networks. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, pages 199–206, New York, NY, USA, 2003. ACM Press.

[11] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *Proc. ACM SIGIR*, pages 267–275, 2001.

[12] J. Ott and D. Kutscher. Drive-Thru Internet: IEEE 802.11b for "Automobile" Users. In *Proc. IEEE INFOCOM*, 2004.

[13] J. Ott and D. Kutscher. Bundling the Web: HTTP over DTN. In *Workshop on Networking in Public Transport*, August 2006.

[14] A. Pentland, R. Fletcher, and A. Hasson. Daknet: rethinking connectivity in developing nations. *Computer*, 37(1):78–83, 2004.

[15] S. E. Robertson. The probability ranking principle in IR. pages 281–286, 1997.

[16] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost communication for rural internet kiosks using mechanical backhaul. In *Proc. ACM MobiCom*, pages 334–345, 2006.

[17] F. Song and W. B. Croft. A general language model for information retrieval. In *Proc. ACM SIGIR 1999*, pages 279–280, 1999.

[18] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *ACM WDTN*, pages 252–259, 2005.

[19] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based search engine for complex queries. In *Proc. Intl. Conf. on Intelligence Analysis*, 2005.

[20] W. H. Yuen, R. D. Yates, and S.-C. Mau. Exploiting Data Diversity and Multiuser Diversity in Mobile Infostation Networks. In *Proc. IEEE INFOCOM*, April 2003.

[21] X. Zhang, J. Kurose, B. N. Levine, D. Towsley, and H. Zhang. Study of a Bus-Based Disruption Tolerant Network: Mobility Modeling and Impact on Routing. In *Proc. ACM Annual Intl. Conf. on Mobile Computing and Networking (Mobicom)*, September 2007.

[22] Y. Zhou, B. N. Levine, and W. B. Croft. Distributed Information Retrieval For Disruption-Tolerant Mobile Networks. CIIR Technical Report IR-412, University of Massachusetts Amherst, 2005.