PeopleNet: Engineering A Wireless Virtual Social Network

Mehul Motani and Vikram Srinivasan
Electrical & Computer Engineering
National University of Singapore
{motani, elevs}@nus.edu.sg

Pavan S. Nuggehalli
Centre for Electronics Design & Technology
Indian Institute of Science, Bangalore
pavan@cedt.iisc.ernet.in

ABSTRACT

People often seek information by asking other people even when they have access to vast reservoirs of information such as the Internet and libraries. This is because people are great sources of unique information, especially that which is location-specific, community-specific and time-specific. Social networking is effective because this type of information is often not easily available anywhere else. In this paper, we conceive a wireless virtual social network which mimics the way people seek information via social networking. PeopleNet is a simple, scalable and low-cost architecture for efficient information search in a distributed manner. It uses the infrastructure to propagate queries of a given type to users in specific geographic locations, called bazaars. Within each bazaar, the query is further propagated between neighboring nodes via peer-to-peer connectivity until it finds a matching query. The PeopleNet architecture can overlay easily on existing cellular infrastructure and entails minimal software installation. We identify three metrics for system performance: (i) probability of a match, (ii) time to find a match and (iii) number of matches found by a query. We describe two simple models, called the swap and spread models, for query propagation within a bazaar. We qualitatively argue that the swap model is better with respect to the performance metrics identified and demonstrate this via simulations. Next, we compute analytically the probability of match for the swap model. We show that the probability of match can be significantly improved if, prior to swapping queries, the nodes exchange some limited information about their buffer contents. We propose a simple greedy algorithm which uses this limited information to decide which queries to swap. We show via simulation that this algorithm achieves significantly better performance. Overall our results demonstrate that PeopleNet, with its bazaar concept and peer-to-peer query propagation, can provide a simple and efficient mechanism for seeking information.

MobiCom'05, August 28–September 2, 2005, Cologne, Germany.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Algorithms, Design, Performance

Keywords

Wireless Networks, Social Networking

1. INTRODUCTION

A popular means of seeking information is asking around. A person may consult her friends, who in turn will ask their friends, and so on, until the information is found. This kind of social networking is very popular in spite of the availability of vast reservoirs of information such as the Internet and libraries. Navigating for information via social networks works better than other methods like the Internet when people are searching for information which is location-specific, community-specific, or time-specific. This is because people are good sources of these types of information, e.g., a good pizza place in Toronto.

Inspired by the benefits of social networking, we conceive a wireless virtual social network which mimics the way people seek information in their social circles. This network leverages the fact that mobile devices are becoming increasingly sophisticated and commonly support multiple network interfaces. For example, a mobile phone, apart from providing access to the cellular network may also allow short-range peer-to-peer connectivity (e.g. using Bluetooth). Development platforms for writing applications and middleware for mobile devices are also readily available. The proliferation of these all-in-one devices can enable seamless information propagation in a virtual social network, allowing automated matching between those who seek information and those who possess it. To see the level of penetration of such devices, we conducted several experiments in the student cafeterias at the National University of Singapore to see how many Bluetooth enabled phones we could find. Using a laptop installed with Linux, a Bluetooth dongle, and the BlueZ protocol stack [1], we scanned for Bluetooth phones in the vicinity. We conducted this experiment during the lunch period over four days with each session lasting 90 minutes. To our surprise, we discovered over 100 unique devices with their Bluetooth radios turned on. This translates to about 1 unique device every 4 minutes.

In this paper, we present an architecture, which we call PeopleNet, to implement a wireless virtual social network. We envision a scenario in which a person possessing or seeking some information enters a query into her mobile device. In the latter, case, we call the query a *request* query, for obvious reasons. In the former case, we call the query a *response* query, because it can potentially match an existing or future request query. Whether it is a response or a request, the query is first propagated via a fixed infrastructure (e.g., cellular) to a few users in a specific geographic region, called a bazaar. The bazaars are pre-determined geographic regions and each bazaar handles only queries of certain pre-determined types. For example, a sports related query will go to a sports bazaar. The users in the bazaar then further spread the query via the peer-to-peer mode. As users are mobile, they are likely to come within radio range of a significant number of other users in the bazaar, resulting in fairly widespread query propagation. Whenever a matching query is found, the user who placed the query is automatically informed of the match (e.g. via email or text message).

PeopleNet can support a variety of services. In fact, any application involving the mutual exchange of information or goods, e.g., buying and selling or dating, can be supported by PeopleNet. For example, suppose that Alice, visiting Boston on a full day business trip, discovers that there is a sold-out Yankees/Red Sox baseball game that night. She wants to see the game but cannot afford to spend the time and effort required in calling around or hunting on the internet. In another part of town, Bob has a ticket but his wife just reminded him that today is their anniversary and he better have made plans or else! Let us see how both Alice and Bob can use PeopleNet to their advantage. Bob places a sell query for the tickets along with an asking price. Alice places a buy query for tickets to the game with her bidding price. Although both queries may have started off in different parts of town, both are forwarded to the sports bazaar over the cellular network. Once in the sports bazaar, these queries are further propagated within the bazaar from device to device. If the queries match, Alice and Bob are notified and they can make the necessary arrangements. Using PeopleNet, both people are able to satisfy their desires by spending very little time and effort.

Storing the kind of information handled by PeopleNet (time, location and community-specific) in centralized databases requires significant time and effort on the part of individual users. From the user perspective, PeopleNet's main advantages are convenience, since your mobile phone is with you wherever you are, and time savings, as illustrated earlier. Web-based content providers like Yahoo!, Amazon, eBay and Craigslist can provide some of these services on the Internet. Establishing such a database requires setting up massive server warehouses, storage area networks, etc., which involves substantial capital investment and incurs considerable recurring costs for operation and maintenance, which may be passed on to users. From the system perspective, PeopleNet is a simple, low-cost and scalable architecture for dynamic information storage and access. Finally, in many emerging markets, mobile penetration is quite high, while the Internet infrastructure is poor. In fact, Business Week reports that India now has more mobile phone than land lines [2]. In such scenarios, PeopleNet is a very attractive mechanism to provide access to information and services for the masses.

There is not much related work in this area. Independent proposals for wireless virtual social networking have been made before [15], [16], [17] and [18]. However, these efforts have focused on exploiting wireless peer-to-peer connectivity to enable specific applications such as dating, etc and provide relatively simple software based solutions. We discuss related work in greater detail in Section 8.

Our contributions in this paper are the following. We first describe the PeopleNet system architecture. In particular, we outline how our system architecture integrates naturally with the cellular infrastructure with minimal software installation. Our system provides for a *distributed geographic database*. It is distributed, because information is stored across different people's devices. In other words *people are the database*. It is a geographic database because queries related to a particular category are directed to a predetermined geographic region via the infrastructure. We measure system performance in terms of (i) probability of a match, (ii) time to find a match and (iii) number of matches found by a query. We describe two simple models called the swap and spread models for buffer management. We first qualitatively argue that the swap model is better and prove this via simulations and analysis. We then analyze the swap model and compute the probability of match. We show that the probability of match can be significantly improved if, before peer-to-peer exchange of queries, the nodes were aware of each others buffer contents. Based on this information, we propose a simple greedy algorithm whose goal is to maximize the number of matches made by a query. We show via simulation that this algorithm achieves significantly better performance.

This paper is organized as follows. In Section 2 we describe the system architecture of PeopleNet. We define some notation and identify the performance metrics in Section 3. In Section 4, we discuss query propagation algorithms and motivate the swap model. In Section 5, we discuss the advantage of the bazaar concept. We study in detail and analyze random swap in Section 6. In Section 7, we introduce the notion of meta-information exchange prior to swapping and show that it can provide significant performance gains. We discuss related work in Section 8. We end the paper in Section 9 by reflecting on PeopleNet and looking at what research issues remain for the future.

2. SYSTEM DESCRIPTION

We assume that users have access to two kinds of network connectivity. Users can communicate over long distances using a fixed infrastructure such as the cellular system. Users can also communicate over short distances with nearby users using peer-to-peer interfaces such as Bluetooth. Users possess (and seek) information sought (and possessed) by other users. The PeopleNet architecture is a lightweight and scalable mechanism for such information exchange. In this paper, we outline the architecture on cellular (for long range propagation) and Bluetooth (for short range propagation). However, PeopleNet can be overlaid on any infrastructure, e.g., WiMax and any peer-to-peer interface, e.g., WiFi.

A given area (say, a city) is divided into non-overlapping regions called *bazaars*. Each bazaar is dedicated to handle certain types of queries placed by users. As noted earlier, queries can be either requests or responses. For example, we could have a sports bazaar, an automobile bazaar, etc. We

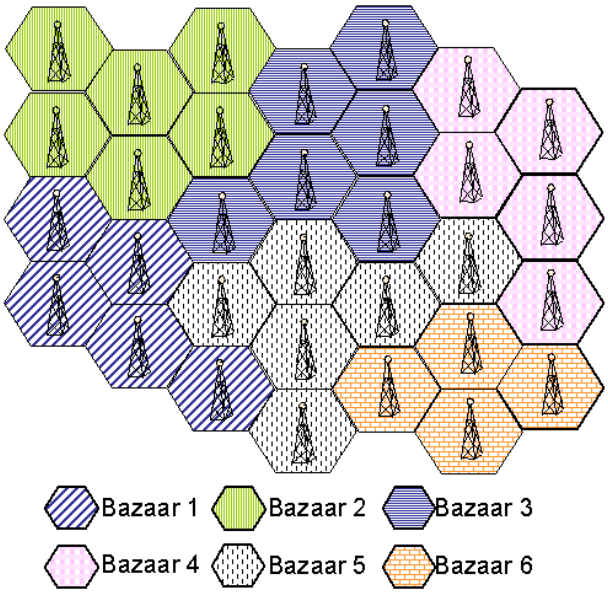

Figure 1: Depiction of the PeopleNet Bazaars as an overlay to an existing cellular infrastructure. The hexagonal cells represent the existing cells. Several cells are clustered to form a bazaar, which are represented by different shading.

note that any user can place a query associated with any bazaar irrespective of where she is located. In other words, a user need not be physically located in the sports bazaar to ask a sports question.

When a user places a query on her device (say a query relating to sports), the query is propagated via the network infrastructure to k randomly selected users in the associated bazaar (sports bazaar). As these k users move around, they propagate the query to other users, who in turn further propagate the query via the peer-to-peer mode. When two matching queries are co-located on a single device, the device automatically informs the respective users about the match via the network infrastructure. For example this could be achieved by sending a text message or email message to the users who initially placed the queries.

The intuition behind creating bazaars is that it speeds up the dynamics of PeopleNet. The key idea is that it reduces the average initial distance between matching queries. Clearly, queries that start off geographically closer have a higher chance of finding each other than those that start off farther apart. This is precisely what creating bazaars does.

2.1 PeopleNet over Cellular Infrastructure

In the cellular context, we can imagine a bazaar spanning the area served by a few base stations which are controlled by a Mobile Switching Center (MSC). A PeopleNet Coordinator (PC) running at the MSC provides the added functionality for PeopleNet. Fig. 1 depicts bazaars as an overlay network on an existing cellular infrastructure.

Information is classified into types (e.g. sports, music, auto) and represented in a well defined format called query. Each PC maintains a look up table mapping query type to the PC controlling the bazaar corresponding to the query type. When a user places a query on her device, the query is first sent to the PC responsible for the bazaar the user is present in. The PC consults its look up table and forwards the query to the PC corresponding to the query type. When a PC receives a query from another PC, it chooses up to k users from the MSC's Visitor Location register (VLR) and transmits the query (wrapped in a text message) to these users. These k users in turn propagate the query to other users in their vicinity via Bluetooth connectivity (in our implementation on Nokia series 60 phones, the RFCOMM usage profile was used). As users move around, queries can spread rapidly in a bazaar. When two matching queries are co-located on a single device, the device automatically informs the respective users about the match via the cellular infrastructure. This could be achieved,for example, by sending a text message or email to the users who initially placed the queries.

We note that the PeopleNet architecture imposes very modest overhead on the cellular system. It is completely transparent at the base station level and requires minimal software installation at the MSC's. Moreover, the PC does not buffer any queries and merely acts as a relaying agent. Since the cellular infrastructure is used only for relaying and not for storage or computation, the system is scalable.

2.2 Query Database

At its heart, PeopleNet constructs a distributed geographic database. The information in this database consists of queries placed by users from their mobile devices. In this paper, the term "query" is used as a generic term to refer to the information that is placed by a user.

This information is organized in a hierarchal structure. The generic query format and a specific example are shown in Fig. 2. The *marker* field is a three bit field. The first two bits indicate whether the query is a *request* or *response*. In our model, responses can only match requests. We note that requests and responses are application specific. For example, a user can place a response query with the opinion of a good pizzeria in Toronto he has just been to. If another user is

looking for a good pizza place in Toronto, he can place a request query for this information. In a buy/sell application, the buy query is a request and a sell query is a response. The third bit of the marker field indicates whether the user placing this query wants to be notified when a match is found for his query. In our earlier examples, the person expressing his opinion on the pizzeria will probably choose not to be notified of a match. However, in the buy/sell application, both the buyer and seller can choose to be notified. The next field provides the user address. The following N fields describe the query in a hierarchical fashion, with level i being higher in the hierarchy than level j when $i > j$. A match occurs between a request and a response when all specified levels of the request match those in the response. The final field is for the user to enter additional information about the query and is not used in the matching process. Figure 2 also illustrates an example of request and response queries in a buy/sell application.

Supposing that there are N levels in the hierarchy, the N-tuple along with the first two bits (i.e., Response & Request) corresponds to a query type. The PeopleNet service provider determines the mappings of query types to bazaars and queries placed by users are forwarded to the appropriate bazaar.

3. DESIGN CHOICES AND METRICS

3.1 Notation and System parameters

We describe here the notation and the metrics of interest to the PeopleNet architecture.

- N - Number of nodes in a bazaar.
- λ - Arrival rate of new queries in the system.
- $2M$ - Total number of query types. For every type i, $1 \leq i \leq 2M$, there is a unique matching type $j \neq i$.
- k - The number of nodes in a bazaar to which the query is sent by the cellular infrastructure.
- B - Size of the buffer at each node.
- L - The number of queries exchanged when two nodes meet.

3.2 System Model for Analysis

We focus on a single bazaar, which is a square grid of size $W \times W$. We assume that there are N nodes populating this world, with each node occupying one of the grid positions. We allow nodes to move in two mobility patterns on the square grid, namely, random walk, wherein nodes can move uniformly to any neighboring grid position, and i.i.d. walk, where a node can move uniformly to any grid position. Each node has a buffer size of B. We assume that there are $2M$ types of possible queries in the system. For every type $i \in 1, \ldots, 2M$, there is a unique matching type $j, j \neq i, j \in 1, \ldots, 2M$. If a query of type i is co-located in a node with a query of matching type, then a match occurs. Time is discrete. In each slot, a new query arrives into the system with probability λ. The type of the query is uniformly distributed between 1 and $2M$. Each query that arrives is randomly distributed to k of the N nodes. The transmission radius of a node is assumed to be $\sqrt{2}$ units and a node can communicate with another node within its transmission radius.

3.3 Metrics of Interest

The primary metrics we are interested in are:

1. Probability of match: Probability that a query finds a match before it leaves the system.
2. Time to match: Average time it takes for a query to find a match given there is a match.
3. Time in system: Average duration for which there is at least one copy of a query in the system.
4. Number in system: Average number of copies of a query in the system as a function of time.
5. Number of distinct matches: Average number of distinct matches that a query finds before it leaves the system.

The quality of service to the end user is governed mainly by the first two metrics. One way to give quality of service guarantees in the PeopleNet architecture is to re-inject a query into a bazaar if it does not find a match. The second and third performance metrics gives us a handle of when this should be done. The last metric is also useful. If a user places a query to sell some article, then she would like as many responses or matches as possible.

3.4 Simulation Setup

We used a software package called RePast [3], to conduct our simulations. Repast was developed at the University of Chicago's Social Science Research Center for creating agent based simulations. Matlab is ill equipped to handle such simulations. Network simulation tools such as ns-2 or Glomosim incorporate many intricate protocol details,which are not required for our simulation models. We do not require TCP or UDP flows, nor complex routing algorithms. Moreover, at the time scales we are considering, MAC level retransmissions and delays can be safely ignored. We also compared the scalability of RePast with SWANS [4], a scalable ad hoc networks simulator developed at Cornell. We found Repast to be far more scalable compared to SWANS for the purpose of our simulations.

4. TO SWAP OR SPREAD?

In this section we will discuss two alternative models for propagating queries in the peer-to-peer mode. We call these the *random spread* and *random swap* models. Let us look at a snapshot of the system when two nodes, n_A and n_B, meet. In random spread, n_A randomly chooses a query from its buffer and send a copy of it to n_B, which puts that query in a randomly chosen location in its buffer. n_B may have to delete an existing query to make space for the new query. Similarly n_B sends across a copy of a query to n_A. In random swap, nodes n_A and n_B choose random buffer location and swap the queries at these buffer locations. Note that the nodes do not retain copies of the queries they transmit.

At first glance, it seems intuitive that in order for a query to find a match quickly, it needs to replicate itself and spread to as many nodes as possible. This will result in the number of copies of the query growing rapidly until it finds a

3-bit Marker Response|Request|Notify	User address	Level 1	Level 2	...	Level N	User Description

101	2405551212	Car	Toyota	Corolla	2001	$7,000	Very good condition, must buy

011	2401212555	Car	Toyota	Corolla	> 2000	$< \$10,000$	Prefer automatic

Figure 2: Generic query format and examples of queries for selling/buying a car with notification.

match. This is precisely what random spread does. However, we argue in this section that in a finite buffer system, the spreading model is actually detrimental to the performance of the system.

Consider a finite buffered system where new queries are constantly coming into the system. Then in steady state every node's buffer will always be full. Therefore, for a node to accommodate an incoming query from a neighbor, it will have to delete some query from its buffer. Queries must also be deleted when new queries arrive at a node.

In the random swap model, we do not attempt to increase the number of copies of the query in the system with time. In random swap, a copy of a query gets deleted only if it is replaced by a copy of a newly arrived query. This implies that each query lives longer in the system, increasing the chances of it finding a match, compared to the random spread model.

Let us compare the two models in between two query arrivals to the system. In random swap, the number of copies of a query in the system remains constant. On the other hand in the random spread model, the number of copies of the system can either increase or decrease.

We compare the two models via RePast simulations with respect to the metrics of interest identified in Section 3. We considered a single bazaar of size $W = 32$, with $N = 30$ users. The buffer size is $B = 3$ and the number of types is $M = 30$. TheF arrival rate of new queries is $\lambda = 0.5$.

From the plots in Figs. 3 and 4, we note something intriguing. First, we see that the time in the system for the random spread model is much smaller than that for the random swap model. However if we compare the expected number of copies of a query in the system as a function of time, we see that the expected values are identical. This essentially implies that the variance for the number of copies of a query at any time t in the random spread model is very high. From Fig. 5, for the probability of a match, it is clear that this variance in the number of copies of a query adversely affects its chances of finding a match. From Figs. 6, we see that the time to match for random swap is marginally worse than random spread. However, from Fig. 7, we see that the number of distinct matches found by a query is much larger for random swap. We found similar behavior for other sets of parameters also.

From our qualitative arguments and our simulation experiments it is evident that the random swap model out performs the random spread model for the PeopleNet architecture. Therefore, for the rest of this paper we focus mostly on the random swap model.

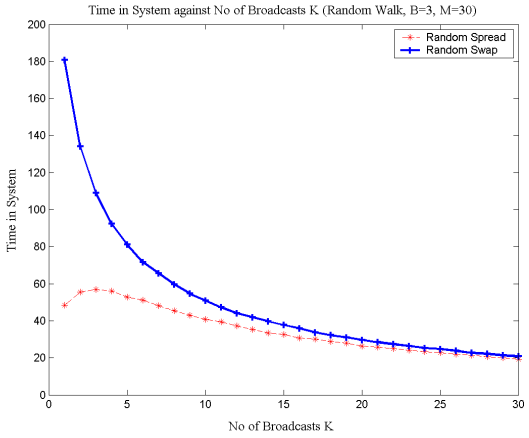

Figure 3: Time in System for Random Swap Versus Random Spread

5. IMPACT OF THE BAZAAR CONCEPT

To illustrate the impact of the bazaar concept, we first prove the following theorem, which shows that for both the random spread and swap models, the expected number of queries of a certain type is constant. This theorem will be useful to understand some of the simulations results presented later in this section.

Recall that we have a city of size $W * W$, with a uniformly distributed population of N users and that there are $2M$ categories of queries that can be posted by the users of the system. Furthermore, the arrival rate of queries to this system is λ and each node has a buffer size of B. Each query that is posted is placed on the devices of k randomly chosen users.

THEOREM 1. *Let $Q_{ij}(t)$ denote the number of type $i, 1 \leq i \leq 2M$ queries in node $j, 1 \leq j \leq N$ at time t and let $Z_i(t) = \sum_{j=1}^{N} Q_{ij}(t)$ be the total number of copies of type i queries in the system. Then, $E[Z_i(t)] = \frac{NB}{2M}, \forall i, \forall t$.*

Proof: Consider the system in steady state, i.e, when all buffers are full. Since

$$\sum_{i=1}^{2M} Q_{ij}(t) \quad = \quad B, \forall j,$$

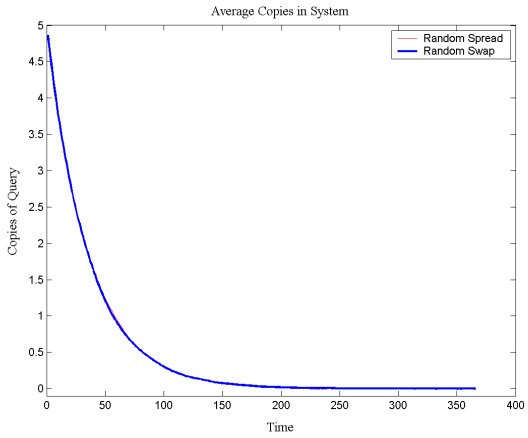

Figure 4: Number of a particular query as a function of time for Random Swap and Random Spread

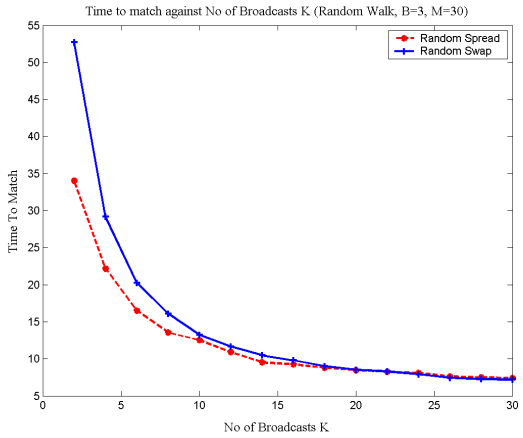

Figure 6: Time to match for random swap versus random spread

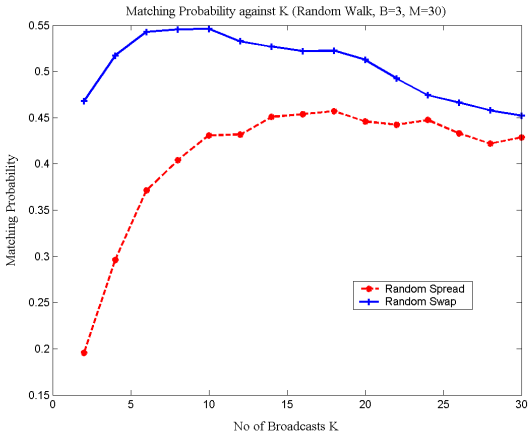

Figure 5: Probability of match for random swap versus random spread

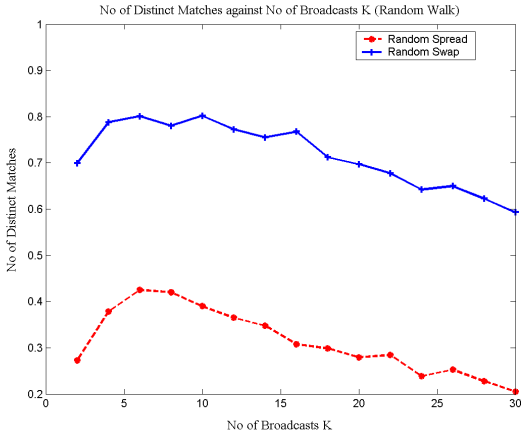

Figure 7: Number of Distinct Matches for Random Swap versus Random Spread

we have

$$\sum_{j=1}^{N}\sum_{i=1}^{2M} Q_{ij}(t) \quad = \quad NB.$$

Interchanging summations and taking expectations,

$$\sum_{i=1}^{2M}\sum_{j=1}^{N} Q_{ij}(t) \quad = \quad NB$$

$$\sum_{i=1}^{2M} E[Z_i(t)] \quad = \quad NB$$

If we assume a buffer management scheme that does not discriminate between different types of queries, then it is clear from symmetry that all the $Z_i(t)$'s are identically distributed. Therefore, we have $E[Z_i(t)] = \frac{NB}{2M}$. ■

Theorem 1 at first glance is disconcerting since the expected number of queries of a certain type in the system at any given time is independent of both k and λ. If we use naive counting arguments, this would seem to suggest that k has no role to play in determining the probability of a match. However, a simple thought experiment illustrates the role of k. Consider a system in which $\lambda = 1$, $k = N$ and $B = 2$ (we require B to be at least 2 for matches to occur). In such a system, any newly arriving query will get flushed out of the system very quickly and if M is large, the probability of a match is very low. Therefore, we see that although the mean is independent of k and λ, the variance is governed by these quantities and contributes to the probability of match. This is clearly illustrated in Fig. 5. In fact we note that there actually exists an optimal value of k which maximizes the probability of a match.

We can now investigate how system performance improves with bazaar planning. Assume, now that we want to split this city into m bazaars of equal size. Then the size of each bazaar will be $\frac{W*W}{m}$, with $\frac{N}{m}$ users in each bazaar on an average. Each bazaar will serve $\frac{2M}{m}$ categories of queries and the arrival rate of queries to each bazaar will be $\frac{\lambda}{m}$. We investigate the performance via simulation. We consider two scenarios, in the first scenario, we have $W = 102$, with $N = 300$, $M = 500$, $\lambda = 0.5$ and $B = 3$. In the second scenario, we have $W = 32$, $N = 30$, $M = 50$ and $\lambda = 0.05$. If we

	$k=5$, $B=3$		
Bazaar Size	Matching Probability	Time to Match	Number of Matches
$N=300$	0.25	280	0.3
$N=30$	0.6	150	0.9

Table 1: Comparison of two bazaars, one with $N=300$, $W=102$, $M=500$ and $\lambda=0.5$, the other with $N=30$, $W=32$, $M=50$ and $\lambda=0.05$.

assume that each grid point is approximately 2m apart, then the values we have chosen translate to a population density of about 7000 people per square km. This is approximately the population density of a moderately sized city such as San Francisco or Singapore. We compare the following metrics via simulation (i) probability of match, (ii) average number of matches made by a query and (iii) time to match. We note from Table 1 that the probability of match and the number of matches made by the system are much higher for the smaller bazaar. From Theorem 1, we observe that the average number of queries in the big and small bazaars of any category is the same. Therefore, on average, queries arriving in both systems have the same number of matching queries in the bazaar. For the smaller bazaar, this number $\frac{NB}{2M}$ is distributed over a smaller number of nodes, which are localized within the smaller bazaar. Therefore, it much easier for a query to find and make multiple matches as it moves through the system. We also see from Table 1 that the time to find a match is much smaller (factor of 1/2) in the smaller bazaar.

Does this imply that each bazaar should be made arbitrarily small? In other words, should we go to the extreme of designing each bazaar with $M=1$?. One important point to note is that in our simulations we assume that each bazaar forms a closed system with all users remaining within the bazaar. In the real word as users move around, they will be in different bazaars at different times. Therefore the mobility patterns of users will actually determine the size of a bazaar. We need to make each bazaar sufficiently large, so that the set of users in a bazaar changes slowly with time.

6. ANALYSIS OF SPREAD AND SWAP

We assume the system model described in Section 3. For completeness, we state bounds on time in system and number in system for the random spread model. We then analyze random swap in detail.

6.1 Results for the Random Spread Model

For the sake of analysis, we approximate the random walk mobility model by the i.i.d walk mobility model, where in each slot a node can move to any random location on the grid with equal probability. In this model, since the transmission radius is $\sqrt{(2)}$, the probability that a node has at least one node within its transmission radius is given by $p=1-(1-\frac{9}{W^2})^{(N-1)}$. We give bounds for the time in the system and average number of copies of a query in the system for the random spread model. We leave out the proofs of the next two theorems for the sake of brevity.

We first define

$$\begin{aligned}\delta &= \frac{\lambda k}{N}\\ C1 &= \frac{\delta+p}{B}-\frac{\lambda p}{B^2}+\frac{\frac{\lambda p(B-1)}{B^2}+\frac{p}{B}}{(N-1)B}-\frac{pN}{(N-1)B}\\ C2 &= \frac{p}{(N-1)B}-\frac{1}{(N-1)B^2}(\frac{\lambda p(B-1)}{B}+p)\end{aligned}$$

THEOREM 2. *Assume that the system is in steady state and a query arrives at time 0, then the expected number of copies of this query at any time t can be bounded by*

$$k(1-C1-kC2)^t \le E[Q_t|Q_0=k] \le k(1-C1-C2)^t$$

Let T_{sys} denote the time that a query stays in the system. Then we can prove the following theorem.

THEOREM 3.

$$E[T_{sys}] \le \min\{k\frac{1-C1-C2}{C1+C2}, \frac{\log(\frac{C1+C2}{k(C1+kC2)})}{\log(1-C1-C2)}\}$$

6.2 Analysis of Random Swap

If two nodes are neighbors, then they each randomly choose a buffer location and swap the queries in these locations. We assume that $M \gg B$ and $k \ll N$.[1] This implies that we can approximate the the distribution of queries in each buffer to be uniform and that we can treat each of the k copies of a query independently. The analysis in this section depends only on the above reasonable assumptions.

THEOREM 4. *For the random swapping model, at any time t, the expected number of copies of a query which arrives at time 0 is given by $E[Q_t]=k*(1-\frac{\lambda k}{NB})^t$.*

Proof: As before, assume that the system is in steady state and a new query arrives at time 0. Let us consider a particular copy u of this query. Then the probability that u is still in the system at the end of time t is given by $\alpha_t=(1-\frac{\lambda k}{NB})^t$. Therefore the probability that there are m copies of the query at the end of time t is given by a binomial distribution and $P(Q_t=m)=\binom{k}{m}\alpha_t^m(1-\alpha_t)^{(k-m)}$. Therefore the expected value $E[Q_t]=k\alpha_t$. ■

THEOREM 5. *The expected time in the system is well approximated by*

$$E[T_{sys}]=\frac{\gamma+\ln k}{\ln(1-\beta)^{-1}}+0.5$$

where, $\gamma=0.577$ is the Euler constant and $\beta=\frac{\lambda k}{NB}$.

Proof: The time in the system is the first time taken for the deletion of all k copies of a query. Let $T_i, \quad i=1,\ldots,k$, be random variables which represent the time at which the i^{th} copy of the query is deleted. Then the time in the system $T=\max_i T_i$. Making the reasonable assumption that $k \ll N$, we assume that the T_i are independent and identically distributed geometric random variables. It is well known that the maximum of a set of geometric random variables does not have a limiting distribution. However, in [5], it was shown that the maximum of k i.i.d geometric random

[1]We believe this will hold in most practical cases.

variables, with parameter $p = 1 - q$, is well approximated by

$$\frac{\ln k}{\ln(1-q)^{-1}} + \frac{\gamma}{\ln(1-q)^{-1}} + 0.5$$

■

THEOREM 6. *The probability of a query finding a match $P(C_k)$ is given by*

$$P(C_k) = 1 - (\frac{r_0 s\beta}{1 - r_1 s})^k \quad (1)$$

where,

$$\begin{aligned} r_0 &= (1 - \frac{1}{2M})^{(B-1)} \\ s &= \{(1-p) + p[(1 - \frac{L}{B})(1 - \frac{1}{2M})^L + \\ &\quad \frac{L}{B}(1 - \frac{1}{2M})^{(B-L)}]\} \\ r_1 &= \left[\frac{\lambda k}{N}(1 - \frac{1}{2M})(1 - \frac{1}{B}) + (1 - \frac{\lambda k}{N})\right] \\ \beta &= \frac{\lambda k}{NB} \end{aligned}$$

Proof: Each slot is divided into two phases, the arrival phase and the swap phase. In the arrival phase, an arrival occurs at a given node with probability $\frac{\lambda k}{N}$. If an arrival occurs at a node, it randomly deletes one of the existing queries from the buffer to accommodate the new query. The node then checks if the new arrival matches any of the existing queries in the buffer. It then moves to the swap phase. In the swap phase, if the node has a neighbor, it randomly chooses L queries from its buffer and swaps it with L queries from the neighboring buffer. Once the L new queries arrive from the neighboring buffer, the node once again checks if these L new queries match any of the existing $B - L$ queries in the buffer.

Assume that the system has been running for a long time and has reached a steady state whereby all buffers are full. Assume that a new query arrives into the system at time zero. Since $k \ll N$, we will assume that the probability of two copies of the same query residing in the same buffer is extremely small and therefore, the statistics of one of the k copies can be studied independently of the other copies $k-1$ copies of the same query. Let us follow copy u of the query.

Consider the following events:
A_n is the event that u has not been deleted by time n and has not found a match.
B_n is the event that u is deleted at time n.
C is the event that u does not find a match.
Then it is clear that

$$C = \cup_{n=0}^{\infty} A_n \cap B_{n+1}$$

However $A_n \cap B_{n+1}$ is mutually exclusive from $A_{n+i} \cap B_{n+i+1}$ for all $i \neq 0$. Therefore, we have

$$P(C) = \sum_{n=0}^{\infty} P(A_n \cap B_{n+1}) \quad (2)$$

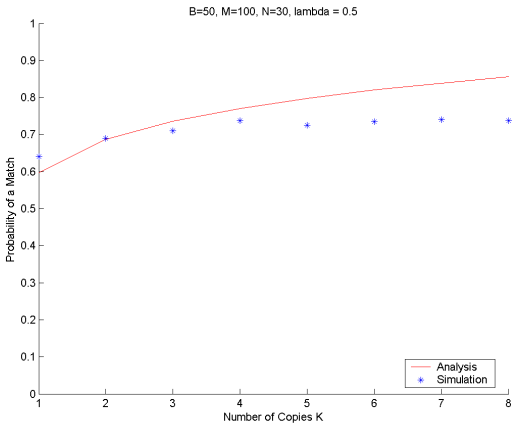

Figure 8: Comparison of Random Swap Analysis versus Simulation with $N = 30$, $W = 32$, $\lambda = 0.5$, $B = 50$, $M = 500$

We can compute the respective probabilities as follows:

$$P(A_0 \cap B_1) = r_0 s\beta \quad (3)$$

where

$$\begin{aligned} r_0 &= (1 - \frac{1}{2M})^{(B-1)} \\ s &= \{(1-p) + p[(1 - \frac{L}{B})(1 - \frac{1}{2M})^L + \\ &\quad \frac{L}{B}(1 - \frac{1}{2M})^{(B-L)}]\} \\ \beta &= \frac{\lambda k}{NB} \end{aligned}$$

Similarly,

$$P(A_n \cap B_{n+1}) = q_0 q_1^{n-1}\beta, \quad n \geq 1 \quad (4)$$

where

$$\begin{aligned} q_0 &= r_0 s \\ r_1 &= \left[\frac{\lambda k}{N}(1 - \frac{1}{2M})(1 - \frac{1}{B}) + (1 - \frac{\lambda k}{N})\right] \\ q_1 &= r_1 s \end{aligned}$$

Putting (3) and (4) together, we can compute the probability of no match for u as

$$\begin{aligned} P(C) &= \sum_{n=0}^{\infty} q_0 q_1^{(n-1)}\beta \\ &= \frac{q_0\beta}{1 - q_1} \end{aligned} \quad (5)$$

Therefore the probability that none of the k copies find a match $P(C_k) = 1 - (\frac{q_0\beta}{1-q_1})^k$ ■

In Fig. 8, we see that this approximation is very accurate for small values of k.

6.3 Match Analysis With a Genie

In this section, we compute an upper bound on what the probability of match could be. When two nodes are neighbors of each other, we assume that there is a genie which looks into both buffers and generates all possible matches.

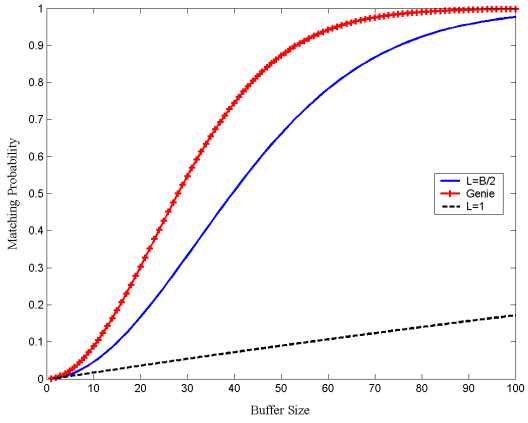

Figure 9: Probability of match versus increasing buffer size. We see that for $L = 1$, the probability of match increases linearly with increasing buffer size, while it increases much faster for $L = \frac{B}{2}$ and for the genie case.

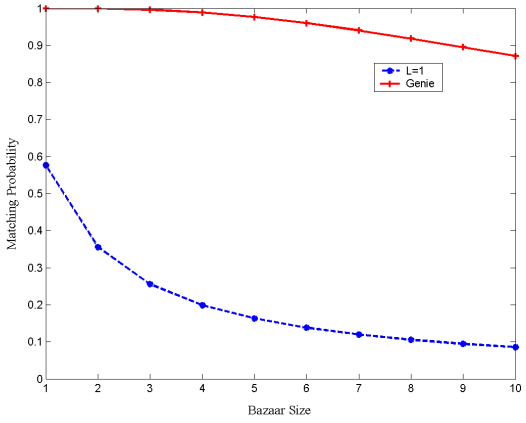

Figure 10: Effect of Increasing Bazaar Size. The number of nodes in the system, M and λ, all scale proportional to the bazaar size. The probability of a match decreases substantially with increasing buffer size. Also the gap in probability of match between the genie case and the case where $L = 1$ is very big.

THEOREM 7. *The probability of match for a query $P(C_{kg})$, with a genie is given by*

$$P(C_{kg}) = 1 - (\frac{r_{0g}s_g\beta}{1 - r_{1g}s_g})^k \quad (6)$$

where

$$r_{0g} = (1 - \frac{1}{2M})^{(B-1)}$$
$$s_g = [(1-p) + p(1 - \frac{1}{2M})^B]$$
$$r_{1g} = \left[\frac{\lambda k}{N}(1 - \frac{1}{2M})(1 - \frac{1}{B}) + (1 - \frac{\lambda k}{N})\right]$$

Proof: As before, we will follow one of the k copies u of a query q which arrives at time 0. If we define the events A_n and B_n as before, then,

$$P(A_0 \cap B_1) = r_{0g}s_g\beta$$
$$P(A_n \cap B_{n+1}) = q_{0g}q_{1g}^{(n-1)}\beta$$

where,

$$q_{0g} = r_{0g}gs_g$$
$$q_{1g} = r_{1g}s_g$$

It is now easy to see that the probability of match with a genie $P(C_{kg})$ is given by

$$P(C_{kg}) = 1 - (\frac{q_{0g}\beta}{1 - q_{1g}})^k$$

■

We can now make a few observations. First, the probability of match in the random swapping model is maximized when $L = \frac{B}{2}$ (recall L is the number of queries swapped when two nodes are neighbors). This is intuitively quite obvious. Second, we note from Fig. 9 that for some values of buffer size, that even with $L = \frac{B}{2}$, there can be fairly large differences with the probability of match of the genie. This difference is much more pronounced for smaller values of L. Also, in Fig. 10, we see that the difference in probability of match between the genie and for small values of L persists across different bazaar sizes.

From a practical perspective, it might not be always feasible to swap $B/2$ queries. This is because the duration for which two nodes are in transmission range is not long enough or because nodes might want to conserve energy. The system, depending on mobility patterns and the user depending on energy constraints might set L to be much smaller than $\frac{B}{2}$. In such a scenario, we see that this *genie gap*[2], can be quite substantial. These observations motivate us to ask the question: are there alternative transmission mechanisms which can increase the probability of match? One obvious solution would be for the nodes to first exchange some meta information, which provides some knowledge of the buffer contents of each node. Based on this knowledge, the nodes could intelligently exchange queries to maximize the various system objectives (probability of a match, number of matches, time to match etc.). We describe and discuss some simple models for buffer exchange based on meta-information.

7. THE IMPACT OF META-INFORMATION EXCHANGE

Suppose two nodes meet and decide to swap queries. The main question is how do the nodes decide how many and which queries to swap. The simplest (and maybe even naive) thing to do is to choose the queries to swap randomly from the buffer. Alternatively, the nodes could employ some intelligence in swapping queries by giving higher priority to certain queries and less to other queries. The method of assigning the priorities to the queries could depend on a variety of factors, including the query lifetime or some future predicted benefit. We note that the naive exchange simply assigns equal priority to all queries and so is a special case of

[2]The difference in probability of match for random swap and the genie case

the weighted exchange. The critical thing to note about the weighted exchange described here is that it only takes into account information known to (or predicted by) the node itself. In other words, it does not factor in information from the other nodes (and their buffers).

In the previous section, we considered the genie-aided case to get a handle on the largest possible matching probability. In this case, the genie is able to generate all possible matches by looking into both buffers. In other words, the genie uses information from both nodes to make the swap decision. Motivated by this, we consider the scenario in which nodes exchange some local information, such as their buffer contents, prior to swapping queries. We call this pre-exchange data *meta-information*. The meta-information will be used by the nodes to make better decisions as to which queries to swap in order to increase the probability of matching queries. In other words, the nodes will jointly decide which queries to swap so that the interaction results in the maximum number of matches. The goal is to maximize system objective 5 described in Section 3.

In this section, we consider the design of meta-information exchange algorithms and their impact on the performance of random swap in the PeopleNet. Suppose two nodes n_X and n_Y meet and exchange L queries. We let B_X and B_Y represent the buffer contents of nodes n_X and n_Y respectively, i.e., B_X and B_Y are vectors of length B whose elements are in the set $\{1Q, 1A, 2Q, 2A, \cdots, MQ, MA\}$ and query type iQ (the request) matches query type iA (the response).

We point out that, in the meta-information exchange interaction, only one of the nodes needs to send over its meta-information, e.g., say n_Y sends to n_X. Once n_X knows the meta-information of n_Y, it can use this information to make a decision as to which queries both nodes will swap. The way that n_X makes its decision is to perform a thought experiment as to what would happen in different scenarios. After n_X makes its decision as to which L queries each node will swap, it simply sends over its queries to n_Y, along with instructions to n_Y as to which queries it should send over to n_X. Summarizing, the meta-information exchange interaction consists of n_Y sending meta-information to n_X, n_X running a thought experiment to decide which L queries each node should swap, each node exchanging those L queries (in a block manner) and finally each node making the appropriate matches in its respective buffer. Note that the meta information only provides type information and not the address of the person who placed the query. We note that an alternative model is to interleave swapping and matching L times. We do not consider this due to additional delays and overhead involved in such a mechanism.

Clearly, the queries that n_X should send to n_Y depends on which queries n_Y has in its buffer and the queries which n_Y should send to n_X depends on which queries n_X has in its buffer. So a clear candidate for the meta-information is the types of queries and number of each type that each node has in its buffer. We let $\tilde{X}$ and $\tilde{Y}$ be the meta- information of n_X and n_Y respectively and note that they are function of their respective buffer contents, B_X and B_Y. As an example, for $B = 9$, if

$$B_X = [1Q, 1Q, 1Q, 1Q, 1Q, 1Q, 2A, 2A]$$

and

$$B_Y = [1A, 1A, 1A, 1A, 1A, 2Q, 2Q, 2Q, 2Q]$$

then,

$$\tilde{X} = [(1Q, 7), (2A, 2)]$$

and

$$\tilde{Y} = [(1A, 5), (2Q, 4)].$$

It is arguable that exchanging meta-information might be too resource consuming. To see that meta-information does not incur much over head, we present some simple computations. In the worst case, the buffer contains queries of distinct types with no repetitions and the meta information consists of the entire buffer content, which is a vector of length B over $2M$ distinct elements. Clearly, the number of bits required to send over this meta-information is $B \log 2M$ bits. Let's compare this to the average size of a query. Consider an average query. It will contain the query type information, which is $\log(2M)$ bits. The address of person placing the query is around 4B, some description of the article being sought, say 250 words = 2KB., and a picture, say 20 KB. If $B = 50$ and $M = 1000$, then the total is around 22KB. On the other hand, the meta information will contain $B \log 2M \approx 1/2\text{KB} << 22\text{KB}$.

We describe a simple greedy algorithm below. When n_X is examining which query types n_Y has, it takes note of which query type has the largest count in $\tilde{Y}$, e.g., suppose it is y_{max} and $v(y_{max})$ is the number of queries of type y_{max}. It then chooses, if possible, the query in its buffer which matches y_{max}. If it does not have a query matching y_{max}, it chooses a query from n_X which matches the second largest count in $\tilde{Y}$, and so on. Similarly, n_X chooses from n_Y the query which matches the type with the largest count in its own buffer, according to $\tilde{X}$, and so on. We call this algorithm *greedy meta-rank*, because it ranks the types according to count and then chooses to blindly match the largest types. Let us continue the example from above. Suppose n_X knows $\tilde{X} = [(1Q, 7), (2A, 2)]$ and $\tilde{Y} = [(1A, 5), (2Q, 4)]$. Then n_X will decide that n_Y should send it a type $1A$ query and it will send n_Y a type $1Q$ query. This will result in 6 matches at n_X and 4 matches at n_Y for a total of 10 matches. Fig. 11 explicitly shows this interaction.

A closer look at the example above will reveal that had n_X decided that n_Y should send it a type $1A$ query and it send to n_Y a type $2A$ query, then there would have been a total of 11 matches, 7 at n_X and 4 at n_Y. This points to a cleverer way for n_X to decide which queries to exchange. The problem with greedy meta-rank is that it can undermine its own greediness in subsequent exchanges, e.g., by exchanging queries it had already matched. The cleverer algorithm simply takes care of these subtle cases so that the exchange does not eliminate matches already made in previous exchanges. The algorithm works as follows. As before, n_X know $\tilde{X}$ and $\tilde{Y}$. It then runs a thought experiment regarding the upcoming query exchange. It first decides who will exchange first by looking at who has the largest number of query types over both buffers and choosing that one (say n_X) to match first. The other buffer (say n_Y) then will choose to send a query which matches the largest type in n_X. Then n_X reorders $\tilde{Y}$ with the new buffer contents, ignoring the query which n_Y will send to n_X, to see which is the query type now has the largest count. n_X then decides to send a query to n_Y which matches the new largest count. One thing to note is that if there is a tie in $\tilde{Y}$ between a type which n_Y sent before and another type, then the other type should be chosen to match. This is because if n_X chooses

Before Swap

Node X		
Rank	Type	Count
1	1(Q)	7
2	2(A)	2
Node Y		
Rank	Type	Count
1	1(A)	5
2	2(Q)	4

X sends 1(Q), Y sends 1(A)
$- - - \rightarrow$

After Swap

Node X		
Rank	Type	Count
1	1(Q)	6
2	2(A)	2
3	**1(A)**	1
Node Y		
Rank	Type	Count
1	1(A)	4
2	2(Q)	4
3	**1(Q)**	1

Figure 11: An example for the greedy meta-rank algorithm. It generates 6 matches at X and 4 at Y for a total of 10 matches.

Before Swap

Node X		
Rank	Type	Count
1	1(Q)	7
2	2(A)	2
Node Y		
Rank	Type	Count
1	1(A)	5
2	2(Q)	4

Y sends 1(A)
$- - - \rightarrow$

Thought Experiment: Step 1

Node X		
Rank	Type	Count
1	1(Q)	7
2	2(A)	2
3	**1(A)**	1
Node Y		
Rank	Type	Count
1	1(A)	4
2	2(Q)	4

X sends 2(A)
$- - - \rightarrow$

Thought Experiment: Step 2

Node X		
Rank	Type	Count
1	1(Q)	7
2	2(A)	1
3	**1(A)**	1
Node Y		
Rank	Type	Count
1	1(A)	4
2	2(Q)	4
3	**2(A)**	1

Figure 12: An example for the smart meta-rank algorithm. It generates 7 matches at X and 4 at Y for a total of 11 matches.

to match the type which n_Y has sent before, it will reduce one match in its own buffer. We call this algorithm *smart meta-rank*.

Continuing with the example from above, Suppose n_X knows $\tilde{X} = [(1Q, 7), (2A, 2)]$ and $\tilde{Y} = [(1A, 5), (2Q, 4)]$. In its thought experiment, n_X will first decide that n_Y will send n_X a query of type $1A$, since n_X has 7 of type $1Q$, while n_Y has 5 of type $1A$. Then n_X will recompute $\tilde{Y}$ as $\tilde{Y} = [(1A, 4), (2Q, 4)]$. Now n_Y has the same number of queries of type $1A$ and $2Q$. n_X could now choose to match either type $1A$ or $2Q$, but it will choose to match type $2Q$ since n_Y has already sent type $1A$ in the previous exchange. So n_X will send to n_Y a query of type $2A$ and this will result in a total of 11 matches. Fig. 12 explicitly shows this interaction.

Let y_{max} be the query type which has the largest count in $\tilde{Y}$ for which n_X has a matching query. Similarly, let x_{max} be the query type which has the largest count in $\tilde{X}$ for which n_Y has a matching query. Let $v(x_{max})$ and $v(y_{max})$ be the number of queries of types x_{max} and y_{max} respectively. Let $\kappa = v(x_{max}) + v(y_{max})$. It can be seen that the maximum number of matches per interaction possible with meta-rank is in the set $\{\kappa, \kappa - 1, \kappa - 2\}$. It can be easily shown that no larger number of matches are possible for any other algorithm. In this sense, meta-rank (for $L = 1$) is an optimal strategy.

We now briefly consider how to do meta-information exchange in the case of $L > 1$. The first thing to note is that the problem gets significantly more complicated. The optimal algorithm (in the sense of maximizing the number of matches per interaction) is a straight-forward, but computationally complex brute-force search. A heuristic algorithm for $L > 1$ is a simple extension of the meta-rank algorithm for the $L = 1$ case. Recall that the exchange decisions are made by X in a thought experiment. In performing this thought experiment, X just uses the meta-rank algorithm L times, each time making one swap decisions. After L times, the L queries to be exchanged are decided and each node swaps these L queries. This is clearly a suboptimal strategy since (in the thought experiment) subsequent exchanges may undo prior matches. However, the algorithm should intuitively perform well since the underlying meta-rank is optimal in the $L = 1$ case.

Table 2 gives simulation results comparing the performance of meta-information exchange with random swap. We have chosen only to show the results for a specific choice of parameters. We have tested the algorithm over various parameters to obtain similar performance gains.

With smart meta-rank, the probability of match improves by about 12% over the naive random swap, the time to match reduces by 40% and the number of matches found increases by eight fold!

In this section, we have demonstrated that exchange of information prior to random swap can significantly improve the performance of PeopleNet. We point out that meta-information exchange also helps random spread but not to the extent that it is better than random swap.

	$k = 5$		
Swap Method	Matching Probability	Time to Match	Number of Matches
Random	0.73	250	0.25
Smart Meta Rank	0.85	150	2.0

Table 2: Comparison of swap methods, with $N = 30$, $W = 32$, $M = 500$, $B = 50$ and $\lambda = 0.5$.

8. RELATED WORK

8.1 Ad-hoc Networks

There has been much research on ad hoc and sensor networks in the recent past. Examples include [6, 7, 8], which are mainly theoretical studies on the capacity improvements and delay-throughput trade-offs that result from mobility, when data is sent from a specific source to a specific destination In contrast, our proposal does not consider the traditional end-to-end communication paradigm. We only assume that there is an information query sent out. A small subset of the population is assumed to have a match/answer for this query and via this "social network" propagation, the query will ultimately find a match.

Moreover, in multi-hop systems there is also the issue of providing incentives to users to relay information for each other. There are a number of papers on ad hoc networks which address this issue [9, 10]. These typically involve mechanisms where users pay each other, or users help each other in relaying traffic in the hope that they will be helped similarly in the future. However implementing such mechanisms in practice is extremely difficult. In contrast, in our architecture, since the cellular infrastructure is utilized, the network operator can provide incentives for users to relay information for each other. Also since the information will be short messages, providing incentives is not expensive.

8.2 Peer-to-Peer Networks

Another related idea is that of peer-to-peer (P2P) networks in the wired/wireless world [11, 12, 13, 14]. In current P2P networks, the requester sends a query which is propagated through a specific community (say Napster) and if a match is found, the response is routed back to the requester and a match is made. If, however, the requester is unavailable or has temporarily disconnected, the match is lost. If desired, the requester must re-initiate the search. Two differences of our network from P2P networks are persistence and semantics. As soon as she initiates the query, the requester can forget about the query. The query, however, does not forget her, and when a match is found, the responder can use a variety of means to contact the requester. In other words, the query is persistent in the network. Additionally, our network allows users to give meaning to the queries that they propagate, through selective computing and forwarding.

8.3 Serendipity, Bedd, MobiLuck and Jambo

Serendipity is a project from MIT Media Labs [15], while Bedd [16], MobiLuck [17] and Jambo [18] are commercial start ups which exploit wireless peer-to-peer connectivity to enable applications such as dating etc. The solutions provided by Serendipity and the other commercial entities are purely software based solutions. They provide simple user interfaces on mobile phones where a user can post his/her profile and place a query. If two users are within Bluetooth range share similar profiles they are immediately alerted. The current implementation of Serendipity, Bedd, MobiLuck and Jambo only do one-hop profile search but it is easy to do multi-hop search and forward of profiles. However, since the queries propagate only through the mobile ad-hoc network, it is not possible to engineer the system to offer sharp service guarantees.

8.4 7DS

7DS [19] is a peer-to-peer resource sharing system, aimed at providing data access to wireless mobile devices. As an example, consider a network of hosts which can communicate over a wireless local area network. Some of these hosts are able to access the Internet via a wireless node, an access point, or Bluetooth. Consider host A who is participating in an Internet access session but has intermittent connectivity to the Internet When A needs access but is not connected, it queries hosts in its proximity for the data. Suppose, hosts B and C receive the query. If they have the data, they can forward it to A. If they do not have the data, but they have Internet access, they can obtain the data and then forward it to A. The emphasis of 7DS is data access provision for wireless mobile users who have intermittent connectivity.

7DS and PeopleNet have in common the notion of propagating queries in a network of wireless mobile devices. The aim of the two networks are different in that 7DS has as its primary goal providing data connectivity to mobile hosts while PeopleNet aims to provide a mechanism for information search and access through a wireless virtual social network. In other words, 7DS provides access to existing electronic resources while PeopleNet taps into both existing electronic resources and virtual resources residing in people themselves. In addition, the architecture of the two are different.

8.5 Shared Wireless Infostation Model

The shared wireless infostation model (SWIM) [20] extends the infostation concept by integrating mobile ad-hoc nodes. The infostation concept, originally proposed by researchers at WINLAB, is based on the idea of using high power base stations (i.e., infostations) to provide high data rate network access to small geographically disjoint areas. The intuition is that users buffer large data until they are in the vicinity of an infostation, thereby incurring a delay. This leads to a natural delay-capacity tradeoff. The SWIM concept extends the infostation idea further by allowing data to travel towards the infostation by hopping through an intermediate mobile ad-hoc network.

Both 7DS and SWIM use an epidemic propagation approach to model the spread of information through the mobile ad-hoc network. In this paper, we have actually shown that in a resource constrained system, epidemic propagation (random spreading) is inimical to the interests of the user. We have shown that a swapping mechanism (random swap) is better.

As with PeopleNet and 7DS, SWIM incorporates the idea of propagating data throughout a network of mobile wireless devices. The main aim of SWIM (as with 7DS) is network access, meaning that devices generate data to be offloaded to the wired network through an infostation. On the other hand, PeopleNet does not require the existence of a "wired network", although PeopleNet can certainly support the activities of SWIM. Rather, PeopleNet exploits the existing distributed database residing in people by constructing a virtual wireless social network.

8.6 Others

The notion of delay tolerant networking (DTN) [21] arises when the connectivity of the underlying network is dynamic. While this is certainly different from traditional networking, it still aims to provide the same functionality, e.g.,

route packets reliably between specific source-destination pairs [22]. In PeopleNet, there is no notion of reliable communication between specific source-destination pairs. In [23], the authors build an experimental delay tolerant network based on pairwise contact between users and collect data statistics. It is clear that a similar methodology could be adopted to understand the dynamics of PeopleNet more deeply.

In [24], the author studies the problem of querying for information over a static wireless sensor network. The network consists of a regular grid of sensors in a unit area, in which each sensor can communicate only with adjacent neighbors. In addition, the nodes do not have directional information either about their neighbors or any other nodes. The problem considered is one in which a querying node transmits a query for some information from an unknown destination node. The basic tool in this work is that of random walks and more specifically the continuous time random walk (or Brownian motion). The paper considers several search strategies in which queries do random walks until they hit the destination. It is clear that the system model described in [24] is markedly different from that of PeopleNet. The query propagation process in PeopleNet is more complex since we consider finite system resources and intelligent matching and query forwarding.

9. REFLECTIONS

The starting point of this paper was the fact that, in spite of the prevalence of extensive databases in libraries and on the Internet and powerful search engines like Google, people still find information (and maybe even prefer to) by merely asking their social contacts. This method of information access does not depend on a robust end to end communication session. Rather, it allows users to place queries and receive responses as they become available. Moreover this means of access also exploits the fact that people are unique sources of time and location sensitive information, i.e., *people are the database*. In this paper we take a first step in exploring how this social navigation for information access can be enabled in a seamless and virtual manner.

The target application we focused on was one of matching queries which complement each other, e.g., compatible persons interested in dating. In this context, we proposed PeopleNet, a simple, low complexity architecture for a wireless virtual social network. We showed how the architecture can be easily integrated into the existing cellular infrastructure. PeopleNet exploits the natural mobility of people and their interactions with other people to pair up matching queries. PeopleNet is also inexpensive and requires minimal maintenance and manpower cost as opposed to centralized solutions which will require a large number of servers and data warehouses.

In the following, we reflect on the PeopleNet architecture and point to interesting ideas and issues that arise from our analysis and simulations.

- **To Swap or Spread?:** One main contribution of this paper is the realization that how queries are propagated in the system affects key performance metrics such as probability to match and time to match. We compared and contrasted two forms of query propagation, i.e, random spreading and random swapping. The former is a natural mechanism based on the idea of epidemic propagation and the intuition that spreading a query throughout the system helps to find a match faster. The latter is based on a different sort of intuition and makes sense in a resource limited system. In a resource unconstrained system (with infinite buffers), random spreading is the right thing to do since it maximizes the exposure that a query gets. However, with random spreading and finite buffers, there is a tension between exposure and query lifetime. This is because for certain queries to spread, others must be deleted. With finite buffers, random swap ensures that the query lifetime is larger than random spread, since queries are only deleted upon new arrivals. In some sense, random swap is opportunistic, since a one-for-one exchange does not endanger any query, but does expose the query to new environments.

- **Meta-Information Exchange:** Another contribution of this paper is the notion of exchanging some meta-information prior to swapping queries. Surprisingly, the motivation for such a practical idea came from the theoretical analysis with a genie who has access to all available information. Our findings regarding meta-information exchange are that it does not consume significant system resources, but does yield significant gains in system performance. From a system point of view, we can view the meta-rank as a buffer management algorithm which tries to maximize a simple reward function, namely the total number of matches in the system per slot. This points to a more generalized approach to buffer management.

- **Generalized approach to buffer management:** Upon meeting, devices need to make two decisions, namely, which queries to exchange with its neighbor and which queries to delete from its finite buffer when it receives new queries from the system. In this paper, we have considered simple buffer management policies. For exchanging queries, nodes either randomly choose queries to swap or they first exchange some meta-information to aid in query selection. We considered a natural candidate for the meta-information in the form of the types and number of queries in each devices' buffer. For deleting queries, we adopted the simple model of randomly deleting queries. A more general approach to buffer management is to construct a weighted utility function for both the deletion and swap processes and optimize it to yield efficient algorithms. For example, we could delete queries based on their priority, which can depend on how long they have been in the system (time to date (TTD)) or how many matches they have found to that point (matches to date (MTD)). The intuition for the former is that older queries ought to be deleted while newer queries should remain in the system. The intuition for the latter is that a query which has found many matches may not need any more matches, while a less matched query does. With respect to swapping queries, the meta-information exchange and swap is already based on maximizing a simple reward function, namely maximizing the number of queries matched to that point. We could easily incorporate TTD and MTD into the swap utility function. Finally, energy considerations, such as the amount of energy available at the node,

could also play a role in the generalized deletion and swap utility functions.

- **System Capacity:** When nodes have finite buffers, it is clear that how the system is loaded has a significant impact on the performance, such as probability of match. In the context of PeopleNet, we can characterize system load by two parameters, namely the arrival rate of new queries into the system, λ, and the number of distinct queries, M. It is clear that as you increase either λ or M, there is an adverse effect on the probability of match. This points to a notion of PeopleNet system capacity region, which we define as the set of (λ, M) for which the probability of match is one, or high enough, say 0.95. Clearly, understanding the system capacity region can help to dimension the system, meaning choosing the bazaar size and the number of query types allowed.

- **System Security:** Security is of paramount importance to ensure the success of applications based on PeopleNet. This is especially so in light of recent concerns about viruses and worms infecting and spreading on cell phones [25]. There is no easy solution to this type of threat, except user education and more powerful virus software. Another aspect of security is data integrity, meaning how does one ensure that queries placed in the system are not modified by intermediate relayers. Finally, privacy of the existing data on one's phone is clearly important to end users. The last two items do admit conceptually simple solutions such as symmetric and public-key cryptography. The challenge, given the limited resources of cell phones and other mobile devices, is computational complexity and code efficiency.

- **Query database:** The PeopleNet system does not make a priori assumptions about the kind of information which people would like to exchange. This creates a need for a query management system which can dynamically adapt to user need. The query management system should support two services. Firstly, it should decide the number and kind of query types supported by PeopleNet. This decision will have to be taken based on the nature of queries received at various PeopleNet coordinator stations over a period of time and will probably require human intervention. Secondly, there should be a mechanism to distribute currently supported query types to users, so that they can effectively use the PeopleNet system. One possible means would be to flood the system periodically with control messages bearing query information.

- **Interbazaar Dynamics:** Another subtle point regarding buffer management concerns inter-bazaar dynamics. So far, we have been concerned with the performance within one bazaar and have made decisions for immediate benefit. PeopleNet, however, consists of a multitude of bazaars and people do indeed move between bazaars. This fact can be taken in to account when making the decision on which queries to swap. Recall that bazaars host certain types of queries. Suppose that a node (in one bazaar) has a query (q) which is hosted in another bazaar. Supposing that node meets another node which is known to travel to that other bazaar routinely, it makes sense for our node to swap q. The idea is to exploit the mobility between bazaars and make swap decisions for future benefit instead of only immediate benefit. This could be incorporated into the generalized weighted utility function described above.

- **Mobility Models:** PeopleNet exploits the natural mobility of people (and their devices) to enable and enhance information access. In our analysis and simulations so far, we have assumed a random walk mobility model. While this may admit some analytical treatment, it may not represent the most likely movement patterns of people using PeopleNet. A closer look at how people actually move about in their daily activities points to a model in which people aggregate in certain places for a certain amount of time before moving on to the next aggregation point. A good example of this is how people act in a shopping mall as they browse from store to store and even sit down to have a coffee. Our future work is to evaluate the performance metrics with a more accurate user mobility model, such as the shopping mall model.

- **Comparison with Alternative Implementations:** An alternative to the distributed architecture of PeopleNet is a completely infrastructure-based system in which users' queries are stored and processed for matches at a central data warehouse. The advantages of such a system are low communication overhead and guaranteed matches (if there are matching queries). However, as we argued earlier, this approach requires a significant capital investment in server warehouses and storage area networks and large recurring costs for operation and maintenance. The PeopleNet architecture offers a tradeoff between the need for centralized storage and processing versus increased communication over the cellular infrastructure and between mobile devices. We note that PeopleNet transmits $O(k)$ more messages on average on the cellular network as compared to the centralized solution. Our analysis indicates that there is an optimum value of $k < N$ which maximizes the matching probability. Moreover, Fig. 5 shows that this optimum value can be reasonably small without degrading performance significantly. The increased energy consumption due to increased mobile-to-mobile communication can be mitigated by using meta-information to reduce the number of queries exchanged. Finally, the device can be put into a sleep state to conserve energy.

- **System Implementation:** We have developed and implemented the client side PeopleNet application on the Nokia series 60 emulator and successfully tested it on Nokia 6600 phones. The Nokia 6600 phone provides Bluetooth for local connectivity It has 6 MB internal shared memory for contacts, text messages, multimedia (images, video and ring tones), and user applications. Our experience with application development in this resource constrained environment has taught us a lot about how to design buffer management and matching algorithms. It has also enabled us to choose the simulation parameters realistically. Suppose that an average query consists of 2KB for text and 20KB

for images. Assuming we have access to 1MB of the internal memory for buffer usage, the buffer can hold a maximum of about 50 queries. We have used buffer sizes of upto 50 in our simulations.

- **Bluetooth Phone Penetration:** PeopleNet depends on people using devices with both long-range and peer-to-peer connectivity. In that context, we conducted several experiments to see the penetration of Bluetooth enabled phones. We set up a laptop installed with Linux, a Bluetooth dongle (with 20 meter range), and the BlueZ protocol stack [1] in the student cafeterias and scanned for active phones. We conducted the experiment during the peak lunch period over four days for 90 minutes each. We discovered over 100 unique devices with their Bluetooth radios turned on, meaning about 1 unique device every 4 minutes. We point out that this experiment only detected phones with their Bluetooth radios turned on. Making the conservative assumption that about half of the people with Bluetooth phones actually turn it on means that our sample contained at least 200 unique devices [3]. Combining this with the increasing support for Bluetooth on mobile devices [26], we believe that PeopleNet is realizable in the near future.

Acknowledgments

We would like to thank Eyal De Lara for his careful reading of the paper and numerous suggestions and comments which substantially improved the paper. We would like to thank Tan Huan Terng for running the Repast simulations, Raghuraman Ramanan for the implementation on mobile phones and Khoo Teck Ping for the Bluetooth penetration experiments.

10. REFERENCES

[1] http://www.bluez.org/
[2] Manjeet Kripalani, "Mobile Telecoms Find Nirvana", *Business Week*, October 4, 2004.
[3] http://repast.sourceforge.net/
[4] http://jist.ece.cornell.edu/
[5] W. Szpankowski and V. Rego, "Yet Another Application of Binomial Recurrence: Order Statistics," *Computing* vol. 43, 1990.
[6] M. Grossglauser and D. Tse, "Mobility Increases the Capacity of Adhoc Wireless Networks", *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 477-486, August 2002.
[7] A. El Gamal, J. Mammen, B. Prabhakar and D. Shah, "Throughput-Delay Trade-off in Wireless Networks", In *Proc. IEEE Infocom 2004*.
[8] M.J. Neely and E. Modiano, "Capacity and Delay Tradeoffs for Ad-Hoc Mobile Networks", *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 1917-1937, June 2005.
[9] L. Buttyan and J. P. Hubaux, "Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks" *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 8 no. 5, October 2003.
[10] V. Srinivasan, P. Nuggehalli, C-F. Chiasserini, R. R. Rao, "Cooperation in Wireless Ad Hoc Networks", In *Proc. IEEE Infocom 2003*.
[11] B. Yang and H. Garcia-Molina, "Efficient Search in Peer-to-peer Networks." In *Proc. ICDCS 2002*.
[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications", In *Proc. ACM SIGCOMM 2001*.
[13] http://www.napster.com/
[14] D. Barkai, "Peer-To-Peer Computing: Technologies for Sharing and Collaborating on the Net," Intel Press, 2002.
[15] http://reality.media.mit.edu/serendipity.php
[16] http://www.bedd.com/
[17] http://www.mobiluck.com/
[18] http://www.jambo.net/
[19] M. Papadopouli and H. Schulzrinne, "Effects of power conservation, wireless coverage and cooperation on data dissemination among mobile devices", In *Proc. ACM MobiHoc 2001*.
[20] T. Small and Z. Haas, "The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way)", In *Proc. ACM MobiHoc 2003*.
[21] http://www.dtnrg.org/
[22] S. Jain, K. Fall, R. Patra, "Routing in a Delay Tolerant Networking", In *Proc. ACM SIGCOMM 2004*.
[23] J. Su, A. Chin, A. Popivanova, A. Goel, and E. de Lara, "User Mobility for Opportunistic Ad-Hoc Networking", In *Proc. 6th IEEE Workshop on Mobile Computing Systems & Applications*, 2004.
[24] S. Shakkottai , "Asymptocis of Query Strategies over a Sensor Network", In *Proc. IEEE Infocom 2004*.
[25] http://www.newscientist.com/article.ns?id=dn5111
[26] http://www.bluetooth.com/products/

[3]Our assumption is based on personal conversations with acquaintances and friends with Bluetooth phones.