MINERVA: Information-Centric Programming for Social Sensing

Shiguang Wang, Shaohan Hu, Shen Li, Hengchang Liu, Md Yusuf Sarwar Uddin, Tarek Abdelzaher
Department of Computer Science, University of Illinois at Urbana-Champaign
Email: {swang83, shu17, shenli3, hl4d, mduddin2, zaher}@illinois.edu

***Abstract*—In this paper, we introduce Minerva; an *information-centric* programming paradigm and toolkit for social sensing. The toolkit is geared for smartphone applications whose main objective is to collect and share information about the physical world. Information-centric programming refers to a publish-subscribe paradigm that maximizes the amount of information delivered. Unlike a traditional publish-subscribe system where publishers are assumed to have independent content, Minerva is geared for social sensing applications where different sources (participants sharing sensor data) often overlap in information they share. For example, through lack of coordination, they might collect redundant pictures of the same scene or redundant speed measurements of the same street. The main contribution of Minerva, therefore, lies in a data prioritization scheme that maximizes information delivery from publishers to subscribers by reducing redundancy, taking into account the *non-independent* nature of content. The algorithm is implemented on Android phones on top of the recently introduced named data networking framework. Evaluation results from both two smartphone-based experiments and a large-scale real data driven simulation demonstrate that the prioritization algorithm outperforms other candidates in terms of information coverage.**

I. Introduction

This paper introduces Minerva; a novel publish-subscribe-based programming system for optimizing information throughput in social sensing applications. Social sensing refers to the act of crowd-sourcing sensor data collection to volunteer participants in exchange for offering data services of interest. A common example is the collection and sharing of traffic speed data by drivers on different streets for purposes of computing speed maps that help plan individuals' commute.

We argue that development of social sensing applications calls for an *information-centric* programming paradigm in that the underlying run-time support is geared at maximizing *information flow*. This, as we show below, is not the same as maximizing *data throughput*. Social sensing applications fit a publish-subscribe model, where the sources involved in data collection are the publishers and the service that computes the quantities of interest is the subscriber.[1] Sources are typically mobile, such as phones or cars, and opportunistic WiFi offloading is used to reduce the cost of data upload (most data plans now charge for 3G/4G data upload, which makes it an unattractive choice for the sensing application). Hence, information propagates from one participant to another when they meet, and is uploaded to the subscriber when a participant has a free upload opportunity. Importantly, unlike the traditional publish-subscribe model, where publishers are independent, social sensing applications typically exhibit information overlap among sources. For example, vehicles waiting in the same traffic jam may collect very similar observations about traffic. Redundancy in data collection thus leads to inefficiency, which motivates a system that can recognize and eliminate the redundancy. Such a system would maximize information flow, as opposed to mere data throughput.

The main contribution of Minerva lies in its information-maximizing data prioritization scheme. It transmits publishers' data in an order that maximizes information flow. Hence, if data transfer is interrupted before all data are transmitted, a notion of information coverage is maximized for the given transfer size. The scheme is suitable for mobile environments where connectivity between nodes may be interrupted due to the nodes' mobility patterns and limited battery capacities. We show that without knowing the data transmission time in advance, which is the common case, no prioritization scheme can guarantee the optimal information throughput. Instead, an approximation bound is derived that is achieved by our prioritization algorithm, making it provably near-optimal.

From an API perspective, Minerva separates application-specific components from application-independent components. We recognize that *information* is a measure that may mean different things to different applications. To keep the information-maximization support in Minerva as application-independent as possible, we ask the programmer to define only one application-specific function per collected content type; namely, *a map function*, which takes a data object as operand and returns its position in a virtual space, called the *information space*, where objects that are closer to each other have more information overlap. When two Minerva nodes meet, they exchange content in an order that maximizes coverage in the information space.

An example map function could be one that places objects (that constitute sensor readings) in a space whose dimensions are the location and time of data capture. Hence, sensor readings at closer locations and times would be closer in the virtual space (which designates that such readings are more redundant). In general, other features may be considered as dimensions of information space. For example, in an application that measures temperature in campus buildings, a more meaningful set of features to consider might be time, building name, and room number. Hence, readings from the same time,

[1]The service also makes the computed results available, but this is done using standard dissemination techniques and is not the focus of this paper.

978-1-4673-5775-3/13/$31.00 ©2013 IEEE

the same building, and the same room number would be more redundant that readings from different times, different buildings or different room numbers. The map function would then map such measurements to a space where feature similarity translates into proximity. Once a map function is defined, information maximization, informally, becomes a problem of selecting points that are far enough apart in the information space, so that they are not redundant. The design of a good map function is an important application-specific problem. To keep the discussion in this paper application-independent, we assume that a good map function, for the application at hand, has already been designed and consider how to use it in order to implement an information-maximizing publish-subscribe service.

Minerva is implemented on top of the recently proposed Named Data Networking (NDN) framework [14]; a network paradigm where data objects are given unique names in a hierarchical name space (reminiscent of a UNIX directory structure), allowing the network to retrieve them by name. By giving collected data objects descriptive names (that encode the features of interest), we allow the map function to be a function of object names only. Hence, Minerva only needs to know data objects names in order to determine, with help of the map function, an information-maximizing transmission order.[2]

We evaluate Minerva using two smart-phone-based experiments as well as a large-scale simulation using the T-Drive dataset collected by Microsoft Research Asia (MSRA) [23]. Evaluation results demonstrate that our prioritization algorithm outperforms other candidates in terms of completeness of information delivered.

The remainder of this paper is organized as follows. We compare our work with the state of the art in Section II and present the system design in Section III. We formulate and solve our problem of maximizing information coverage in Section IV. The implementation and evaluation for our proposed solution are discussed in Section V. Finally, we conclude the paper in Section VI.

II. State of the Art

Social sensing attracted much attention in the research community since it was introduced in Burke *et al.* [5]. Examples of early services include CenWits [12], CarTel [13], BikeNet [6], PoolView [9], and GreenGPS [8]. *Application-specific* redundancy-eliminating sensing services, such as PhotoNet [18] and CARE [20], were proposed in earlier social sensing literature. In contrast, our paper is the first to offer a *general application-independent* architecture that maximizes information *coverage*, while allowing customization (via the map function) to the specific application. We further design a novel data prioritization algorithm that is proven to maximize coverage subject to an approximation bound. The work is broadly related to the area of Information Centric Networking (ICN), investigated in recent years [10], [11], [17].

Liu et al. proposed a QoS-heterogeneous prioritization algorithm [15], to allow data packets with deadlines to be transmitted first in order to increase the possibility of offloading them faster. Previous efforts exist on redundancy elimination in networks including application-level [21] and packet level [4] techniques. Their work only try to eliminate redundant data, but do not consider the information carried in the data. For example, if two files have the very similar content, such as traffic speed measurements of the same street block at the same time, but different names, their work will consider these two files as different ones. However, these two files actually are redundant in information. Our work focus on eliminating redundancy in information.

We exploit the Named Data Networking (NDN) framework because it offers significant simplifications in the implementation of information-centric programming. NDN is recently proposed as a future Internet architecture, introduced by Van Jacobson [14], [24]. Since then, several papers investigated aspects of this framework such as suitability to ad hoc networking [16] and naming for mobile environments [19]. Our work is the first in proposing a programming API for social sensing applications that uses the NDN framework to simplify the maximization of information coverage.

III. System Design

This section describes the detailed system design. We first present the system model for social sensing applications, then explain the programming framework based on this model.

A. System Model

Figure 1 depicts the system model for our proposed social sensing applications. Mobile devices participate in these applications by generating and sharing sensory data, which are stored locally and uploaded to a backend server via opportunistic WiFi offloading. Hence mobile devices serve as *publishers*, and the backend server acts as the *subscriber*. Opportunistic peer-to-peer communication might also be enabled to allow information to transparently propagate from one participant to another when they meet, in hopes of finding an offloading opportunity to the server faster. We adopt the NDN framework [14], thus data generated by users are identified by descriptive names.

B. Programming Framework

The programming support in Miverva is straightforward. Minerva provides a publish and a subscribe interface. Additionally, the application provides a callback function (one per content type), called map(), that takes as operand the name of a content object of a particular type and returns the corresponding position and coverage in a virtual information space. The position and coverage of a data point are used to compute its priority in transmission that maximizes information coverage in a resource-constrained environment. Objects

[2]The only element of NDN used by Minerva is that data has hierarchical names. In principle, Minerva can run on top of any system that offers a hierarchical, globally unique naming scheme.

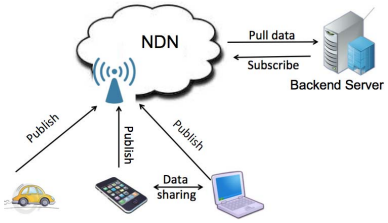


Fig. 1. System Model

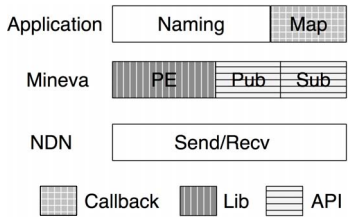


Fig. 2. Programming Framework

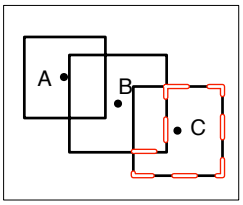


Fig. 3. Illustration of coverage and marginal coverage with 2 features.

are transmitted in the order of largest increase in marginal coverage as discussed in detail in Section IV.

As shown in Figure 2, an operational system would consist of three different layers: the application layer, the Minerva layer, and the network layer.

The application layer would take care of application-specific functions, such as content naming and publishing. Maintaining uniqueness of names is an application-specific concern not addressed in this paper. A fully specified name refers to a unique item. Names can also be partially specified to designate a collection of items that share a common name prefix. In Minerva, publishers and subscribers refer to content collections by name when expressing availability of or interest in content.

In general, applications that use our publish-subscribe system own "subdirectories" in the global name space. For example, an application called *GreenGPS* might own the subdirectory "/root/GreenGPS". The application might publish multiple types of content. Each part could start with "/root/GreenGPS/content-type". Following the content type in the name comes a listing of content attributes of relevance to the map function. A type-specific map function can therefore parse the name to determine the attributes, and compute the coordinates of the object in virtual space accordingly. For example, an object might be called "/root/GreenGPS/content-type/location/time/filename", where the location and time are the features of the data object.

In addition to the publish and subscribe functions, Minerva internally has a core function, *PE*, short for Prioritization Engine. *PE* reflects our optimization algorithm described in Section IV to compute priorities for data objects such that they are transmitted in an order that contributes to maximum coverage.

The underlying network layer provides the communication functions across a network. In our implementation, we use NDN as the underlying network layer. Our solution does not require any changes to the standard NDN library (developed by PARC). Thus, it is general enough to be compatible with other existing NDN applications.

In the next section, we describe in detail the prioritization algorithm.

IV. Information-Maximizing Prioritization

In this section, we first introduce the definition of information coverage and formulate the information coverage maximizing problem. Then, we present the design and analysis of our algorithm.

A. Information Coverage

Data collected in social sensing applications are not independent; they exhibit correlations as discussed in the introduction. For the purpose of theoretical problem formulation, we assume that each data point covers a region in the *information space*, referred to as the *data coverage* region, defined below.

Definition 1 (Data Coverage): Suppose that there are k features of the data collected in a social sensing application. The Cartesian product[3] of domains of the k features forms a k-D *information space*. Any data point X with coordinates $\langle x_1, x_2, \cdots, x_k \rangle$ covers an interval I_j centered at x_j on the j-th dimension, where $1 \leq j \leq k$. The *coverage* of X is $\mathcal{C}_X = I_1 \times I_2 \times \cdots \times I_k$, where $\times$ is the Cartesian product.

Please note that data of different applications might have different coverage intervals. Given a particular application, the notion of coverage is usually clear. For example, when measuring temperature in a corn field, the "coverage" in the time dimension might be, say, 10-20 minutes, since weather does not noticably change in such a short time. Similarly, coverage in space might be 200-300 meters, since these distances are small enough not to dramatically affect temperature. Hence, given a temperature measurement at some time and location it can be assumed to remain valid for the entire coverage interval (in space and in time).

By Definition 1, the coverage of a data point is a k-D box as illustrated in Fig. 3. The coverage of a dataset $\mathbb{S}$ is defined as $\mathcal{C}_{\mathbb{S}} = \bigcup_{S \in \mathbb{S}} \mathcal{C}_S$. The coverage of the intersection (*resp.* union) of two datasets $\mathbb{S}_1, \mathbb{S}_2$ is defined as $\mathcal{C}_{\mathbb{S}_1 \cap \mathbb{S}_2} = \mathcal{C}_{\mathbb{S}_1} \cap \mathcal{C}_{\mathbb{S}_2}$ (*resp.* $\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{S}_2} = \mathcal{C}_{\mathbb{S}_1} \cup \mathcal{C}_{\mathbb{S}_2}$).

We define the *marginal coverage* of a data point X *w.r.t.* a dataset $\mathbb{S}$ in Definition 2.

Definition 2 (Marginal Coverage): The marginal coverage of a data point X *w.r.t.* a dataset $\mathbb{S}$ is the region in the information space covered by X but not covered by $\mathbb{S}$, *i.e.*, $\mathcal{MC}_{X|\mathbb{S}} = \mathcal{C}_X - \mathcal{C}_{\{X\} \cap \mathbb{S}}$.

As illustrated in Fig. 3, the area surrounded by the dashed red line is the marginal coverage of data point C *w.r.t.* the dataset $\{A, B\}$. By definition, $\mathcal{MC}_{X|\emptyset} = \mathcal{C}_X$.

We define the value of the coverage of a data point in Definition 3.

Definition 3 (Coverage Value): The coverage value of a data point X in a k-D information space is the *size* of its

[3]The Cartesian product of two sets A and B is a set C, such that $C = \{\langle x, y \rangle \mid x \in A, y \in B\}$. Similarly, we can define the Cartesian product of k sets.

k-D coverage region, defined as $\mathcal{V}(\mathcal{C}_X) = \prod_{i=1}^{k} I_i$, where I_i is the coverage interval in the ith dimension as in Definition 1.

For example, if $k = 2$, the value of the coverage of a data point is simply the area of its coverage region in the information space. Similarly, definitions of the coverage value of a dataset and the marginal coverage value of a data point *w.r.t.* a dataset follow.

B. Problem Definition

A common goal of social sensing is to gather *information* that is as complete as possible. One trivial solution is that when a connection is established between two participants they sync all data, and when connecting to the backend server, a participant offloads its entire local data. However, due to the mobility and resource constraints (*e.g.*, energy), it is not always possible to sync or offload the entire dataset in a single transmission. Thus, in each transmission session, we aim to maximize the marginal information coverage value of the subset of data that can be transmitted, referred to as the MAXINFO problem.

In the rest of this paper, we shall assume that all data objects of the same type are of the same size. This is a common assumption in sensing applications. For example, in the context of a particular navigation application, all GPS readings have the same format and size. Similarly, in the context of a particular environmental sensing application, all temperature and humidity, measurements have the same format and size. In general, if the data format for sensory measurements is fixed, then all data records have the same size. This assumption simplifies terminology, allowing us to represent connection duration by a corresponding number of transmitted objects. It can be easily generalized to arbitrary object sizes simply by weighting each object by its size. Assuming same size objects, the the MAXINFO problem is formulated as follows:

*Problem 1 (*MAXINFO*):* Suppose that there is a dataset $\mathbb{S}_1$ (*resp.* $\mathbb{S}_2$) on the data receiver (*resp.* the data provider). MAXINFO is to determine an order based on which the receiver should pull data from the provider such that for any data transmission size the receiver's information coverage is maximized. In other words, let $\mathbb{R} \subset \mathbb{S}_2$ with cardinality n is the dataset pulled by the order, then $\forall n, \forall \mathbb{T}, |\mathbb{R}| = |\mathbb{T}| = n, \mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{R}}) \geq \mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{T}})$.

Unfortunately, the unpredictability of the duration of each transmission session makes it *impossible* to find an order that is optimal for any n, which can be proved by a counter example as illustrated in Fig. 3. When $n = 1$, the optimal order is to select data object B first, since its coverage value is the highest. When $n = 2$, the optimal order is to select data objects A and C first, which conflicts with the optimal order when $n = 1$. Hence, we need to quantify the best one can do to approximate the optimal MAXINFO solution when one does not know connection duration in advance.

We first define the optimal solution OPT for MAXINFO to be an offline coverage-maximizing solution that assumes knowledge of the cardinality n in advance. It will constitute a theoretical upper bound, since such knowledge is generally not available online. Note that, as illustrated above, OPT may return different optimal orders for different values of n. Let us define the approximation ratio of an online solution A as follows:

Definition 4 (Approximation Ratio): Consider a dataset $\mathbb{S}_1$ (*resp.* $\mathbb{S}_2$) on the data requester m_r (*resp.* the data provider m_p). Let solution A of MAXINFO represent a fixed priority order for data object to pull from m_p. Let $\mathbb{A}^n \subset \mathbb{S}_2$ denote the subset of data transmitted from m_p with cardinality n during the transmission session. Let $\mathbb{OPT}^n$ denote the subset output by OPT with n known in advance. The approximation ratio of A is

$$\tau = \min_{\forall n, 0 \leq n \leq |\mathbb{S}_2|} \frac{\mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{A}^n})}{\mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{OPT}^n})}.$$

Please note that for any fixed n, when $\mathbb{S}_1 = \emptyset$,the MAXINFO is exactly the weighted *Max n-Cover* problem [7].

Theorem 1: [7] If Max n-Cover can be constructively approximated in polynomial time within a ratio of $(1-1/e+\epsilon)$ for some $\epsilon > 0$, then $NP \subset TIME(p^{O(\log \log p)})$, where p is the cardinality of the set (as $|\mathbb{S}_2|$ in Definition 4).

Theorem 1 directly implies that achieving a better approximation ratio than $(1 - 1/e)$ for MAXINFO is NP-hard. Thus, we have the following Corollary.

Corollary 1 (Approximation Bound for MAXINFO*):* Achieving approximation ratio $(1 - 1/e + \epsilon), \forall \epsilon > 0$ for MAXINFO is NP-hard.

C. Greedy Algorithm

In this section, we outline our prioritization algorithm. The idea of the algorithm is to give higher transmission priority to data with larger marginal coverage value *w.r.t.* the dataset at the receiver side.

Algorithm 1 Prioritization Algorithm

Input: Two sets $\mathbb{S}_1$ and $\mathbb{S}_2$
Output: An order of elements in $\mathbb{S}_2$

1: Set $\mathbb{T} \leftarrow \mathbb{S}_1$
2: FIFO Queue $\mathcal{Q} \leftarrow \{\}$
3: **while** $\mathbb{S}_2 \neq \emptyset$ **do**
4: $\quad X \leftarrow \arg\max_{X \in \mathbb{S}_2} \mathcal{V}(\mathcal{MC}_{X|\mathbb{T}})$
5: $\quad \mathbb{T} \leftarrow \mathbb{T} \cup \{X\}$, $\mathbb{S}_2 \leftarrow \mathbb{S}_2 - \{X\}$, $\mathcal{Q}$.inqueue(X)
6: Return $\mathcal{Q}$

We now prove that the approximation ratio of Algorithm 1 is $(1 - \frac{1}{e})$.

Lemma 1: For any $n \leq |\mathbb{S}_2|$, if $\mathbb{R}$ is the set of the first n elements of the queue output by Algorithm 1, we have

$$\mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{R}}) \geq (1 - 1/e) \cdot \mathcal{V}(\mathcal{C}_{\mathbb{S}_1 \cup \mathbb{R}'}), \forall \mathbb{R}', |\mathbb{R}'| = |\mathbb{R}| = n,$$

where $\mathbb{S}_1$ (*resp.* $\mathbb{S}_2$) is the dataset at the data receiver (*resp.* provider) side.

The proof of Lemma 1 is similar as that in [7], except that in [7] $\mathbb{S}_1 = \emptyset$. Hence, we do not repeat the proof here. Lemma 1 directly implies the following theorem.

Theorem 2 (Approximation Ratio): The coverage value of the transmitted set based on the order output by Algorithm 1 is a $(1 - 1/e)$-approximation of MAXINFO.

Note that the apprximation ratio matches the approximation bound in Corollary 1.

D. Transmission Protocol Design

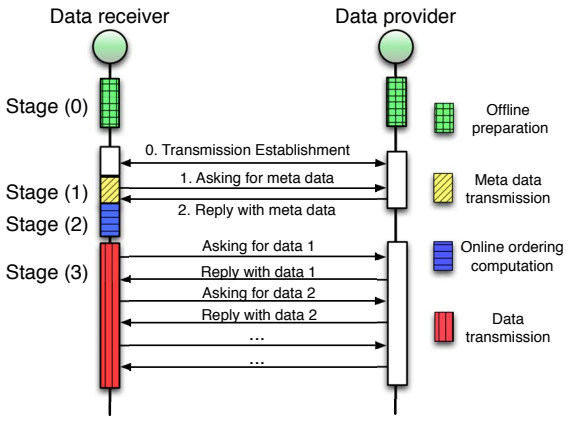


Fig. 4. Transmission protocol illustration.

In this section, we present the transmission protocol, as illustrated in Fig. 4. Since we target mobile platforms, the transmission is occurs in a disruption-tolerant (DTN) fashion; a device shares its data with a peer or offloads to a backend server when the corresponding connection is established. Thus, each transmission session (in which case we say the device is *online*) is followed by an idle session (when we say the device *offline*).

Each transmission session consists of three stages; (1) meta data transmission, (2) online ordering, and (3) data transmission. Stages (1) and (2) are the transmission overhead. Metadata, here, refers to the (information space) coordinates of data objects available at each node. In an NDN-based implementation, these coordinates can be computed from data names (using the map function). Hence, in our implementation, metadata refers to data object names. The idea being that data object names are generally much shorter than the data objects themselves. Hence, it makes sense to exchange the names first, then let each node specifically request from the other the named data objects it deems complementary (i.e., not redundant with) its own.

Before transmission, an offline preparation operation that generates the meta data needs to be carried out to reduce the overhead of the online ordering computation. We now present the offline preparation algorithm and online prioritization algorithm in detail as follows.

1) Offline Preparation: The offline preparation stage outputs a meta data file which contains a list of data names as well as the overlap set of each *heavily informative* data point as described below. (The overlap set of data X contains any data Y *s.t.* $\mathcal{C}_Y \cap \mathcal{C}_X \neq \emptyset$.)

Consider two data points X and Y. If the coverage of X greatly overlaps with that of Y, then after X has been transmitted, Y carries little extra information. Thus, we introduce a constant threshold $\beta > 1$, such that, when the distance between X and Y is smaller than $\frac{1}{\beta}$ we only need to consider one of them (*w.l.o.g.*, say X) in the online prioritization algorithm. The other is put in the lowest priority bin for transmission. We apply this rule repeatedly until no more points can be assigned lowest priority. The surviving points (not assigned lowest priority) are called *heavily informative* data points. Note in particular that if X has no neighbors Y whose distance from X is smaller than $\frac{1}{\beta}$, then X will never be assigned lowest priority and is therefore a heavily informative data point. The heavily informative dataset contains all the data points like X.

Our offline preparation algorithm determines for each new data point whether or not it is heavily informative. If it is heavily informative, it adds the point to the metadata file to be exchanged on contact with another node. It is incremental in the sense that when new data points arrive, we do not need to redo the preparation for old data. The psuedocode for the offline preparation algorithm is presented below.

Algorithm 2 Preparation Algorithm

Input: Existing dataset $\mathbb{S}$, existing meta data file, newly arrived dataset $\mathbb{T}$

Output: Updated meta data file

From meta data, get the heavily informative dataset $\mathbb{H}$ of $\mathbb{S}$
Sort $\mathbb{H}$ based on the lexicographical order of data coordinates in the k-D information space
$\mathbb{D} \leftarrow \emptyset$, $\mathbb{N} \leftarrow \emptyset$
for $\forall S \in \mathbb{T}$ **do**
  Use binary search to find its overlap set $\mathcal{O}_S \subseteq \mathbb{H}$
  if $\exists E \in \mathcal{O}_S$, *s.t.* $S \simeq E$ **then**
    $\mathbb{D} \leftarrow \mathbb{D} \cup \{S\}, \mathbb{T} \leftarrow \mathbb{T} - \{S\}$, **continue**
  Add S to the overlap set of any element in $\mathcal{O}_S$
  Insert S into $\mathbb{H}$ *s.t.* $\mathbb{H}$ remains sorted
  $\mathbb{N} \leftarrow \mathbb{N} \cup \{S\}$
Add the name following by the overlap set of each data in $\mathbb{N}$ to the front of the meta data file
Append names of data in $\mathbb{D}$ to the end of meta data file
Return the meta data file

In our online ordering stage, we only need to consider the heavily informative dataset. The parameter β controls the cardinality of the set of highly informative data. The smaller β is, the smaller the cardinality of this set. In practice, we can use β as a knob to trade-off the accuracy and the time efficiency in the online prioritization as discussed below.

2) Online Prioritization: Online prioritization is described in Algorithm 3. After metadata (i.e., names of heavily informative objects) have been exchanged, the receiver calculates the marginal coverage value of each data object in the highly informative set *obtained from the data provider*, and puts the these values into a max heap. Then, it sends a request for the data object D poped from the max heap, and at the same time updates the marginal coverage value of each data object in the overlap set of D and do the standard heapify. This process continues until the max heap is empty, then, if the connection is still up, the receiver starts to pull data that not in the highly informative set in FIFO order.

Algorithm 3 Online Prioritization Algorithm

Input: The highly informative dataset $\mathbb{S}_1$ on the receiver side which is sorted based on the lexicographical order of data coordinates in the k-D information space, meta data from the data provider

Output: The transmission order (represented by a FIFO queue $\mathcal{Q}$)

1: Initiate a Max-heap $\mathcal{H}$ that stores and sorts data points according to their associated values
2: **for** Each heavily informative S in $\mathbb{S}_2$ **do**
3: Use binary search to get its overlap set $\mathbb{T} \subseteq \mathbb{S}_1$
4: Calculate $v_S = \mathcal{V}(\mathcal{MC}_{S|\mathbb{T}})$
5: $\mathcal{H}$.add(S, v_S)
6: Initiate a FIFO queue $\mathcal{Q}$
7: **while** $\mathcal{H} \neq \emptyset$ **do**
8: $X \leftarrow \mathcal{H}$.popMax()
{At the same time send request X to the data provider.}
9: $\mathcal{Q}$.enque(X)
10: For each data S in X's overlap set $\mathcal{O}_X$, update $v_S = \mathcal{V}(\mathcal{MC}_{S|\mathbb{T}\cup\mathcal{Q}})$, and update the heap $\mathcal{H}$
11: Append all other data in $\mathbb{S}_2$ to $\mathcal{Q}$
12: Return $\mathcal{Q}$

3) Overhead Analysis: The time complexity of Algorithm 3 is $O(m \log n + mn^{\frac{k-1}{k}} + m\beta^k \log m)$, where n is the number of highly informative data in $\mathbb{S}_1$, m is the number of highly informative data in $\mathbb{S}_2$, and β^k is the size bound of a overlap set as stated in the offline preparation section. In the worst case, $n = |\mathbb{S}_1|$ and $m = |\mathbb{S}_2|$, however both of them are related to the parameter β. Thus β serves as a knob to trade-off prioritization accuracy and computational efficiency in our algorithm. We will further study the parameter β in the overhead evaluation.

V. Evaluation

In this section, we study the performance of Minerva. We first describe the experimental setup and evaluation methodology of Minerva for both real-phone based experiments and simulations using the T-Drive dataset collected by MSRA [3]. Then we present evaluation results.

A. Experimental Setup and Methodology

Minerva is designed for social sensing applications with resource constraints. Thus, we need to evaluate two aspects of the system: (i) the overhead of data prioritization, and (ii) the application performance, measured in terms of application-level metrics; namely, information coverage. The following methodology was used for evaluation:

- *Data prioritization overhead:* In order to measure overhead under a wide set of workloads, we generate synthetic load (i.e., synthetic data to be transmitted) that can be easily parameterized to represent a large set of relevant properties. These properties include the size of the data set, the dimensionality of the data, and the degree to which the data is redundant. We then test the overhoad of prioritizing such data on real phones.
- *Application-level performance:* In order to evaluate application-level performance metrics (namely, coverage), we find an actual data trace of a participatory sensing application. We then compare coverage when Minerva is used and when other data transmission schemes are used, given a simplified network simulator.

To accomplish the above, we implemented Minerva on Google Galaxy Nexus smartphones [2], equipped with a 1.2 GHz dual-core CPU, 1GB RAM, and 802a/b/n Wifi radio with Android OS 4.1. Minerva is implemented using the Java language on top of PARC's CCNx prototype software [1]. The overhead study is conducted in an outdoor environment on real phones. The application performance study uses a real-world taxi trace dataset, the T-Drive dataset [3], [22], [23], which contains the GPS trajectories of $10,357$ taxi cabs during the period from February 2nd to February 8th, 2008 in Beijing. The total number of points in this dataset is about 15 million and the total distance of the trajectories is around 9 million kilometers. The trajectories covered are shown in Figure 5.

Fig. 5. The data considered in the simulation.[4]

B. Overhead of Minerva

A key goal in the overhead study is to understand the overhead of data prioritization (which includes the overhead of metadata transmission for purposes of computing priorities) for a wide set of workloads. Hence, synthetic data is used. In this study, workload generation does not attempt to mimic characteristics of any specific application. Rather, it attempts to investigate overhead under a broad range of conditions that affect it. These include, the size of the data set (in terms of the number of objects), the dimensionaity of data, and the degree of redundancy among data items.

To explore the effect of data dimensionality, we generate data by (uniformly) sampling from a k-dimensional box of unit size in the information space, where k is a configurable parameter that represents the number of features (i.e., information space dimensions) considered in the Minerva prioritization algorithm. We use a unit box and uniform sampling because it allows us to easily control the degree of data redundancy by tuning the value of coverage interval associated with individual data points. We focus on overhead only (as opposed to the time it takes to send the data objects). Hence, we measure the

[4]This figure is borrowed from [3].

overhead of sending metadata and computing priorities only. At the of this overhead all objects are properly prioritized and ready for transmission. The results of the overhead study are shown in Table I.

The data set parameters considered in the table are (1) the number of features, (2) the number of data points, (3) the coverage interval per point (and hence degree of redundancy of data), and (4) the value of $1/\beta$ as defined in Section IV. The more data points are considered, the more computation is needed for data prioritization. Similarly, the larger the coverage interval of individual data points, the higher the redundancy (or the probability that two data points overlap in coverage), and hence the higher the computation overhead of redundancy-minimizing prioritization. The value of $1/\beta$ is a parameter of our algorithm that indicates its tolerance of imprecision in redundancy minimization. The higher the $1/\beta$, the more approximate the prioritization, and the lower the data prioritization overhead, as discussed in Section IV. Rows 1-12 of Table I show the total time in metadata exchange and prioritization between two android phones that have the same amount of data points on both phones. Rows 13-16 show the corresponding overhead when an android phone uploads data to a backend server with a 3.10GHz CPU and 8GB RAM.

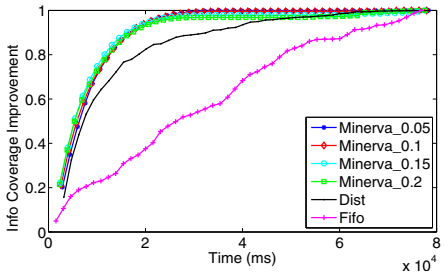
TABLE I
OVERHEAD OF MINERVA

index	dataset features (# dim, # points, interval, $1/\beta$)	overhead(s)
1	2 , **250** , 0.05 , 0.1	0.321 ± 0.165
2	2 , **500** , 0.05 , 0.1	0.837 ± 0.208
3	2 , **750** , 0.05 , 0.1	3.070 ± 1.240
4	2 , **1000** , 0.05 , 0.1	7.205 ± 2.579
5	2 , 500, **0.01**, 0.1,	0.339 ± 0.071
6	2 , 500 , **0.03** , 0.1	0.582 ± 0.104
7	2 , 500 , **0.05** , 0.1	0.837 ± 0.208
8	2 , 500 , **0.07** , 0.1	1.667 ± 0.320
9	2 , 500 , 0.05 , **0.15**	0.700 ± 0.140
10	2 , 500 , 0.05 , **0.2**	0.626 ± 0.145
11	**3** , 500, 0.05 , 0.1	0.257 ± 0.093
12	**4** , 500, 0.05 , 0.1	0.204 ± 0.040
13	2, (10000, 500), 0.01, 0.1	0.076 ± 0.044
14	2, (100000, 500), 0.01, 0.1	1.152 ± 0.093
15	2, (1000000, 500), 0.01, 0.1	4.773 ± 0.537
16	2, (1000000, 500), 0.01, 0.2	0.727 ± 0.104
17	Wifi connection establish time	2.002 ± 0.106

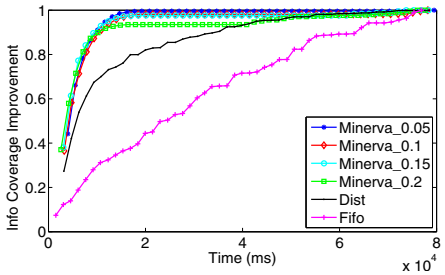
From the table, we observe that the overheads increase as the number of data points increase (rows 1-4) or as the coverage interval increases (rows 5-8), but decrease as $1/\beta$ increases (rows 9-10), which corroborates expectations. When the number of data points is 1000 on both phones (row 4), Minerva takes about seven seconds to prioritize and all objects, which is unacceptable. The overhead drops off sharply with size of the data set (rows 1-3). With 500 objects (row 2), the overhead is less than one second, which is tolerable. Measurements reported in the next section show that one can send roughly $250K$ bytes during that time. Hence, if objects are $250K$ bytes long, the overhead of prioritizing 500 objects is roughly equal to the transmission time of one object. In other words, the prioritization overhead is acceptable as long as the individual objects are sufficiently long.

The effect of the number of features on overhead is shown in row 11 and 12 in the table. It can be seen that the overhead decreases in higher-dimensional spaces (all else being equal), because for the same number of data points and the same coverage interval, higher dimensionality means a sparser space, and hence less redundancy, and less overhead for resundancy-minimizing prioritization.

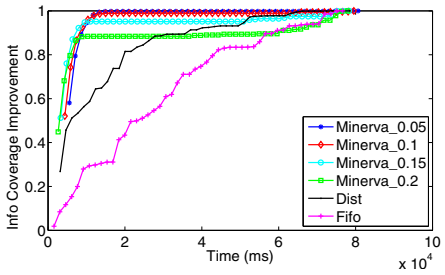
The overheads when data is offloaded from a mobile device to the backend server are shown in Row 13 to 16. The number of data points is set to 500 on the participant side, and the number of data points on server side is set to be $10,000$, $100,000$, and $1,000,000$. We observe that as the number of data points increases on the server side, the overhead grows. We can also observe that the slope of overhead increase becomes smaller when the number of data points at the server side becomes larger. The reason is that the coverage improvement grows submodularly; when the server already got a large enough amount of data, the probability that a new data point is redundant is close to 1.



(a) Coverage interval 0.063



(b) Coverage interval 0.077



(c) Coverage interval 0.089

Fig. 6. Performance of Minerva with different coverage intervals and $1/\beta$ values in phone-based experiments with synthetic data.

Next, we study (in Fig. 6) the coverage achieved with Minerva transmissions between two smartphones using the synthetic data. The number of data points is set to 500 on both phones, and the coverage interval is set to be 0.063,

0.077, and 0.089 (thus, in expectation, one data point overlaps with 2, 3, and 4 other points respectively). For each interval value, we plot the coverage improvement using Minerva (with $1/\beta \in \{0.05, 0.1, 0.15, 0.2\}$), Dist (a distance-based data selection algorithm used in PhotoNet [18]) and FIFO. The x-axis is the time in milliseconds that starts right after the connection is established. The y-axis denotes the normalized information coverage at the receiver side, where a coverage of 1 is equivalent to transmitting all sender data. Remember that the objective of prioritization with Minerva is maximize the coverage of transmitted data (i.e., achieve close to 1 coverage as early as possible during transmission).

From Fig. 6, we observe that Minerva outperforms the other algorithms in general in that it achieves higher coverage earlier on. The larger the coverage interval of individual objects, the better Minerva performs. From the figure, to get 80% coverage, Minerva uses 10 (8 and 5 *resp.*) seconds for a coverage interval of 0.063 (0.077 and 0.089 *resp.*). Dist uses around 20 seconds to achieve 80% coverage, while FIFO takes more than 50 seconds. Minerva coverage also decreases somewhat with increased $1/\beta$ due to the approximation involved.

C. Large-scale Trace-based Evaluation Results

In order to test application-level performance with Minerva in large-scale applications, we emulate a hypothetical *real-time street view* application. This application applies social sensing in a future where vehicles are equipped with cameras. Participants are requested to send pictures from their car's cameras to the base station when they encounter an access point. These pictures are then used on the server to provide a real-time view of city streets on demand.

In this evaluation, we run simulations on real taxi traces in Beijing (the T-Drive dataset). We consider data within the area from latitude 39.5^oN to 40.5^oN and from longitude 116^oE to 117^oE, where most data resides. In the simulation, we assume that there are two sinks that collect data for the backend server as indicated by the two stars in Fig. 5. They are located in two relatively busy roads, where cars can have a higher chance of offloading their data. Cars are assumed not to share data with each other. In our simulation, if the distance between a car and a sink is smaller than 200 meters, we assume that the car can offload data. We assume that each data point (a picture) is 1M bytes long.

We determine the transmission duration of each car by examining its speed when it enters the transmission range of a sink; the speed can be estimated from the time and location information of the latest GPS samples.

In order to obtain a realistic WiFi transmission rate, we conduct a small experiment where we use a smartphone to send a long file over Wifi in an outdoor environment to a desktop in our lab. The average data transmission rate is found to be approximately 250KB per second, and is set accordingly in the simulation.

In our simulations, we consider two features per data sample, the latitude and the longitude. The coverage interval for each data point is set to 50, 100, and 500 meters respectively in subsequent simulations. We simulate for 10 hours' data (1,500,000 points) and assume at the very beginning the server does not have any data.

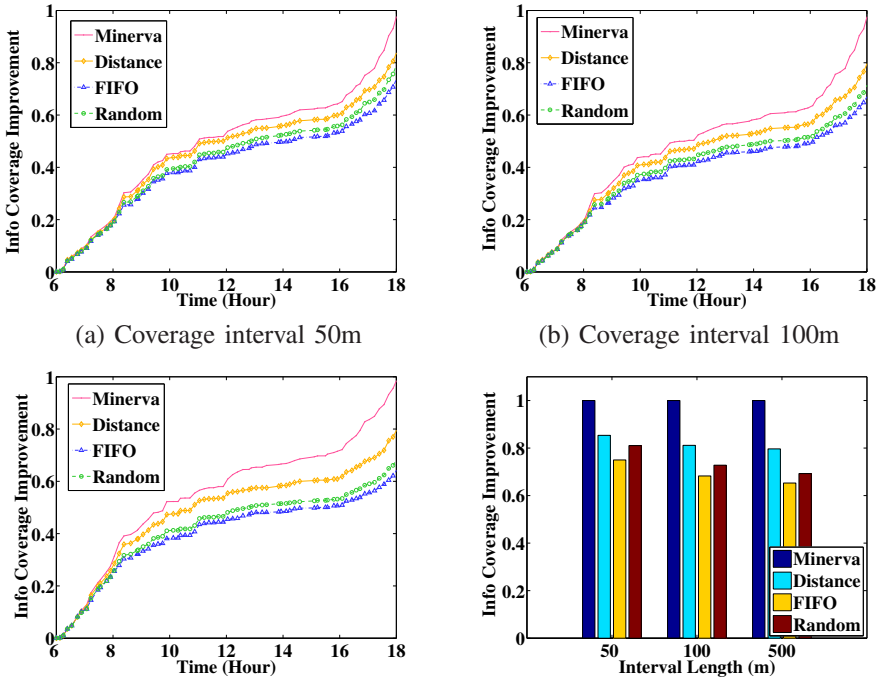
The results are shown in Fig. 7. From Fig. 7, we observe that at the beginning of data collection, the four algorithms compared yield similar performance. Minerva is slightly worse than the others due to its overhead. After collecting data for one hour, Minerva begins to outperform the others. The reason is that Minerva is designed for eliminating redundant data. At the beginning, there is little redundancy since the server has only a limited amount of data. Hence, the overhead is not paying off in terms of application-level metrics. As more data is collected, more redundancy is exploited by Minerva. Eventually, Minerva outperforms all other algorithms in terms of attained coverage. Note that the slope of the coverage curves changes over the course of the day (where the x-axis is time of day). This is because more data is collected during rush hour, roughly between 6-8am and 4-6pm. In summary, evaluation shows that redundancy-minimizing data prioritization has promise in terms of improving coverage of the physical environment in social sensing. However, overhead remains an issue, which reduces applicability except where sensed objects are sufficiently large (e.g., multimedia objects).

VI. Conclusions

In this paper, we presented Minerva; an information-centric programming paradigm and toolkit for social sensing. Minerva is geared for social sensing applications, where different sources (participants sharing sensor data) often overlap in information they share, which distinguishes them from a regular publish-subscribe system where publishers are independent. One contribution in this paper lies in an algorithm for maximizing information delivery from publishers to subscribers taking into account the *non-independent* nature of content. We analytically prove that our transmission prioritization scheme is within a constant approximation ratio from a "clairvoyant" optimal solution. We develop a programming toolkit to embody our prioritization scheme. The scheme is implemented over NDN. To the best of our knowledge, this is the first paper that leverages the benefits of NDN in maximizing information coverage for social sensing applications. Evaluation results show that our algorithm outperforms other candidates in terms of information coverage for large data objects.

Acknowledgement

Research reported in this paper was sponsored by the Army Research Laboratory, DTRA grant HDTRA1-10-1-0120, and NSF grants CNS 1040380 and CNS 09-05014, and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.



(a) Coverage interval 50m (b) Coverage interval 100m

(c) Coverage interval 500m (d) Final coverage improvement

Fig. 7. Performance of Minerva with different coverage intervals in large-scale real-world GPS trace simulations.

REFERENCES

[1] CCNx prototype software. `http://www.ccnx.org`.

[2] Google galaxy nexus. `http://www.google.com/nexus`.

[3] User guide of t-drive data. `http://research.microsoft.com/pubs/152883/User_guide_T-drive.pdf`.

[4] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker. Packet caches on routers: the implications of universal redundant traffic elimination. In *ACM SIGCOMM Computer Communication Review*, volume 38, pages 219–230. ACM, 2008.

[5] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and MB Srivastava. Participatory sensing. 2006.

[6] S.B. Eisenman, E. Miluzzo, N.D. Lane, R.A. Peterson, G.S. Ahn, and A.T. Campbell. Bikenet: A mobile sensing system for cyclist experience mapping. *ACM Transactions on Sensor Networks (TOSN)*, 6(1):6, 2009.

[7] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[8] R.K. Ganti, N. Pham, H. Ahmadi, S. Nangia, and T.F. Abdelzaher. Greengps: A participatory sensing fuel-efficient maps application. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 151–164. ACM, 2010.

[9] R.K. Ganti, N. Pham, Y.E. Tsai, and T.F. Abdelzaher. Poolview: stream privacy for grassroots participatory sensing. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 281–294. ACM, 2008.

[10] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in content-oriented architectures. In *Proc of SIGCOMM Workshop on ICN*, 2011.

[11] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM, 2011.

[12] J.H. Huang, S. Amjad, and S. Mishra. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 180–191. ACM, 2005.

[13] J.H. Huang, S. Amjad, and S. Mishra. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 180–191. ACM, 2005.

[14] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N.H. Briggs, and R.L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 1–12. ACM, 2009.

[15] H. Liu, A. Srinivasan, K. Whitehouse, and J.A. Stankovic. Mélange: Supporting heterogeneous qos requirements in delay tolerant sensor networks. In *Networked Sensing Systems (INSS), 2010 Seventh International Conference on*, pages 93–96. IEEE, 2010.

[16] M. Meisel, V. Pappas, and L. Zhang. Ad hoc networking via named data. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*, pages 3–8. ACM, 2010.

[17] G. Tyson, N. Sastry, I. Rimac, R. Cuevas, and A. Mauthe. A survey of mobility in information-centric networks: challenges and research directions. In *Proceedings of the 1st ACM workshop on Emerging Name-Oriented Mobile Networking Design-Architecture, Algorithms, and Applications*, pages 1–6. ACM, 2012.

[18] M.Y.S. Uddin, H. Wang, F. Saremi, G.J. Qi, T. Abdelzaher, and T. Huang. Photonet: A similarity-aware picture delivery service for situation awareness. In *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, pages 317–326. IEEE, 2011.

[19] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, and L. Zhang. Data naming in vehicle-to-vehicle communications.

[20] U. Weinsberg, A. Balachandran, N. Taft, G. Iannaccone, V. Sekar, and S. Seshan. Care: Content aware redundancy elimination for disaster communications on damaged networks. *Arxiv preprint arXiv:1206.1815*, 2012.

[21] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H.M. Levy. On the scale and performance of cooperative web proxy caching. *ACM SIGOPS Operating Systems Review*, 33(5):16–31, 1999.

[22] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324. ACM, 2011.

[23] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108. ACM, 2010.

[24] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J.D. Thornton, D.K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. Technical report, PARC, Tech. report ndn-0001, 2010.