Zoom: A Multi-Resolution Tasking Framework for Crowdsourced Geo-Spatial Sensing

Thanh Dang   Wu-chi Feng   Nirupama Bulusu
Portland State University, Oregon, USA
Email: dangtx,wuchi,nbulusu@cs.pdx.edu

Abstract—**As sensor networking technologies continue to develop, the notion of adding large-scale mobility into sensor networks is becoming feasible by crowd-sourcing data collection to personal mobile devices. However, tasking such networks at fine granularity becomes problematic because the sensors are heterogeneous, owned by the crowd and not the network operators. In this paper, we present Zoom, a multi-resolution tasking framework for crowdsourced geo-spatial sensor networks. Zoom allows users to define arbitrary sensor groupings over heterogeneous, unstructured and mobile networks and assign different sensing tasks to each group. The key idea is the separation of the task information (what task a particular sensor should perform) from the task implementation (code). Zoom consists of (i) a map, an overlay on top of a geographic region, to represent both the sensor groups and the task information, and (ii) adaptive encoding of the map at multiple resolutions and region-of-interest cropping for resource-constrained devices, allowing sensors to zoom in quickly to a specific region to determine their task. Simulation of a realistic traffic application over an area of 1 sq. km with a task map of size 1.5 KB shows that more than 90 % of nodes are tasked correctly. Zoom also outperforms Logical Neighborhoods, the state-of-the-art tasking protocol in task information size for similar tasks. Its encoded map size is always less than 50% of Logical Neighborhood's predicate size.**

I. Introduction

With the convergence of technology developments in sensor-equipped mobile smart phones, geospatial web interfaces, high data rate 3G and 4G wireless cellular standards, and open WiFi networks, the notion of adding large-scale mobility and coverage into sensor networks by *crowdsourcing* sensor data collection to mobile phones is becoming feasible. Applications of such crowdsourced geospatial sensing systems encompass monitoring of traffic, weather, air and noise pollution, endangered birds in oil spills, and health studies.

As shown by the Mobile Millennium Project at Berkeley [2], the MetroSense project at Dartmouth [5] and Participatory Sensing at UCLA [4], it is possible to create a sensor network that (i) is heavily mobile, and (ii) has sufficient incentives for people to actively participate in sensing. The crowdsourced sensor networks, however, are highly *unstructured* in the sense that the sensor nodes are mobile, heterogeneous, owned by individuals, and are operated at will depending on the individual's incentive to contribute sensor data. Thus, it is difficult to focus the sensed bit (data) over the geographic region in a fine-grained way. For example, querying sensor data over the whole network might end up getting the amount of data that is correlated with the density of sensors (i.e. most data comes from where most cars or people are) but not necessarily relevant to the phenomena being studied.

To maximize the sensor data utility, it is important to provide *structure* over the existing unstructured sensor network. The sensor network application operator should be able to configure the network to control the who, what, and where of sensing in arbitrarily non-uniform, fine-grained ways across a large sensing region. As an example, for an engineer trying to understand arterial (non-highway) traffic flow, finer-grained sensing of location, emissions, or speed may be required along problematic bottleneck areas. While there are high level approaches to determine where measurements should be sampled [9], there is not a systematic way to enable the network operator to program arbitrary nodes in the network to perform the required tasks at a large geographical scale. In effect, these crowdsourced sensor networks can collect a fixed amount of data at any given time. It is important to be able to place the bits geographically that make the largest impact to the application.

In this paper, we propose Zoom, a framework to support location-based sensing over large geographic regions for crowdsourced heterogeneous sensor networks. Our design goals for the Zoom framework are (i) simplicity, (ii) efficiency, (iii) support for heterogeneous devices, (iii) support for unstructured and mobile networks, and (iv) adaptation to resource constraints. The key idea of our approach is to use map-based encoding of tasks, essentially a task map that represents the who, what, and where of sensing. Each pixel in the map represents a small square region that corresponds to a square region in the physical world (i.e. the where). The pixel value specifies the who and what. This allows us to specify an arbitrary group of sensors with arbitrary sensing area depending upon the application. Furthermore, maps can be encoded at multiple resolutions, allowing for finer-grained control of the who, what, and where to sense.

The contributions of this paper include:

- Zoom, the first map-based framework for configuring a mobility-based, large geospatial region, sensor network: The innovation of Zoom is the use of maps to simultaneously encode both the task identification and the task location. Zoom is essentially different from existing approaches in that it makes task creation and assignment visually intuitive. It supports heterogeneous

sensor platforms by decoupling the task information from the task implementation (code), and by unifying the interpretation of task encoding using a well-known standard image encoding and decoding technique.
- Three techniques to adapt Zoom to resource impoverished embedded mobile platforms: The first technique is to encode maps at different resolutions to support platforms with different resources. The second technique is to allow sensor nodes to select a specific region to be encoded, reducing the size of the map. Finally, Zoom can encode a map into independent blocks and let sensor nodes quickly locate the relevant block to decode. The three resource adaptation techniques give Zoom the flexibility to work with a wide range of sensor platforms.
- Evaluation of the framework using simulation: With maps of size only 35 KBytes, Zoom can define a task over a region of 600 sq. km. with only 2 % error in identifying road segments. Simulation of a realistic traffic application over an area of 1 sq. km. with a task map of size 1.5 KBytes shows that more than 90 % of the nodes are configured correctly.

The rest of this paper is organized as follows. Section II compares the new features and benefits provided by Zoom to previous work in tasking sensor networks. Section III covers key design assumptions. The overall design of Zoom is elaborated in Section IV. Section V evaluates Zoom and explores the various factors impacting its performance, demonstrating that Zoom is more efficient in tasking crowdsourced, mobile, geospatial sensor networks compared to the state-of-the-art approaches. Finally, we conclude in Section VI.

II. Related Work

Crowd-sourced sensing is rapidly gaining popularity with the rapid proliferation of sensor-equipped smart phones and is being used for large-scale geospatial sensing applications as diverse as noise pollution monitoring and traffic monitoring. Zoom is intended to task such devices, which span heterogeneous hardware platforms and sensing capabilities. Tasking a sensing network is a form of macroprogramming, which basically provides a high level programming model for the network that abstracts away the details of individual nodes [17]. In other words, it derives local actions at individual nodes or groups of nodes to achieve the desired global behavior of the network [3]. At its core, macroprogramming involves defining groups of nodes and assigning each group a task to perform. We now review grouping mechanisms in previous work.

Grouping mechanisms developed in previous work can be classified into two main categories — *attribute-based* and *rule-based* (Figure 1). Cougar [24] and TinyDB [15] are examples of the attribute-based approach, wherein the sensor network is typically considered as a database. Nodes and the data are named. SQL-like queries are used to task nodes to report data. The node group is defined within the query. This is a preliminary approach for data collection and can only support limited in-network processing tasks. It is also difficult to define multiple arbitrary groups of nodes. Hood [22], Abstract Regions [21], and Logical Neighborhoods [16] are examples of the rule-based approach, wherein groups are often defined by a set of rules. A node whose state, including sensing capability, location, or sensing data, satisfies the rules is a member of the defined group. The rules may be defined based on physical parameters or logical parameters. For example, in Logical Neighborhoods [16], logical nodes are specified by attributes and logical neighborhoods are specified by the set of nodes satisfying a constraint on the nodes' attributes. The constraint is basically a predicate to determine if a node belongs to the logical neighborhoods. The rule-based approach offers greater flexibility and capability in creating groups than the attribute-based approach. It is still challenging to define arbitrary groups of nodes using either the attribute-based approach or the rule-based approach. Zoom addresses this challenge by using a *map-based* approach. Essentially, the whole sensor network can be represented as a spatial map and groups can be defined arbitrarily on the map. This approach allows Zoom to task arbitrary groups of sensors with varying granularity.

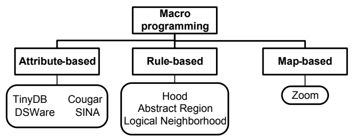


Fig. 1. Macroprogramming abstraction categories

Figure 2 depicts an alternate view of prior work in macroprogramming sensor networks, encompassing two main categories – supporting homogeneous networks and supporting heterogeneous networks. Many early macroprogramming paradigms [22], [15], [21], [24] were designed for homogeneous networks. Only a few attempts [16], [1] support heterogeneous and mobile networks. Interestingly, no prior work adapt the macroprogramming model to different platforms with a range of memory, computation, and power capabilities, a distinct feature of heterogeneous networks. Zoom not only supports heterogeneity and mobility but also provides three adaptation techniques for different sensor platforms.

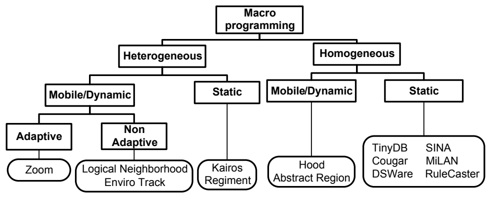


Fig. 2. Taxonomy of macroprogramming for sensor networks

Table I summarizes features provided by different programming abstractions. Only Zoom offers support for heterogeneity, mobility, and adaptation.

Also noteworthy and pertinent to our work are protocols for sensor nodes to efficiently download new code over the air and ensure safe and secure installation of new code. Typical examples of such protocols are Deluge [10] and its secure descendant Seluge [14]. These protocols assume that the nodes

Abstraction	Spatial Scope	Heterogeneity	Mobility	Adaptation
Hood	local	No	Yes	No
Abstract Region	local	No	Yes	No
Logical Neighborhood	Regional	Yes	Yes	No
Zoom	Global	Yes	Yes	Yes

TABLE I
A COMPARISON OF FEATURES ACROSS MACROPROGRAMMING MODELS.

already know what task they are supposed to perform and focus solely on network-wide dissemination. Zoom fills this gap. Its purpose is to explicitly tell the nodes what task they are supposed to do in the first place.

Finally, we have not found any work that defines a data structure for encoding task information with grouping information in sensor networks. We believe Zoom is the first to explicitly address this problem and use maps to encode tasks and group information for large geographical sensing networks.

III. Design Assumptions

Before describing the assumptions underpinning Zoom, we clarify our notion of *task information* and *task implementation*. A *task* is a set of operations which one or more nodes need to perform to accomplish a high-level objective. For example, one task could be to collect sound measurements at 5 KHz and reporting the measurements to a predefined base station. Another task could be to track a moving vehicle. The description of a task is task information, which can be represented by a unique *task index* (e.g. task index 1 is tracking temperature contours). A list of task indices for popular sensing tasks should be specified for every node. For a new task index, a node may need to obtain the appropriate *task implementation*. The task implementation is platform specific and could be some parameters for an existing program, a binary update of an existing program, or even a new binary image that is required to perform the task.

The design of Zoom assumes:

- *Location-Awareness:* Nodes in the network know their own location, obtained from GPS receivers or other localization methods. This a reasonable assumption because our target platforms are hand-held devices, which often have built-in GPS.
- *Dissemination protocols:* There exist protocols to disseminate task information in the network. This assumption is reasonable because there are a wide range of dissemination protocols for networked sensing applications. Dissemination can also be done via radio broadcasts.
- *Code Update:* There exists protocols for a node to download the task implementation to perform a specific sensing task. The goal of Zoom is to help nodes determine their task. This can be done either via peer to peer communication or by downloading code from a nearby base station using WiFi or 3G networks.
- *Push Communication:* A simple approach for a node to determine its task is to periodically *poll* a predefined server for the task information. This approach allows the server to assign exact tasks to individual nodes. The drawback is that every node has to actively poll the server, incurring high bandwidth usage. An alternative approach is *push* based - let the server periodically broadcast the network-wide task information to the network. Although a message containing the task information for all nodes will have a larger size compared to a message containing the task information for a single node, this approach allows nodes to disseminate the task information within the network and keep the whole network updated using fewer transmissions.

IV. Zoom Framework

In the following section, we describe the overall architecture of Zoom, the Sensor Task Interchange Format (STIF) for encoding group and task information, and finally three resource adaptation techniques for resource-poor sensor platforms.

A. Zoom Architecture

Figure 3 shows an overview of the Zoom framework, consisting of two main components — task encoding and task decoding. Task encoding is performed at the back end where an operator can arbitrarily define geographical regions and assign a sensing tasks to each region. The task indices with the location information can be represented as a task map. A location on the map corresponds to a real physical location. The pixel value at a particular map location is the corresponding task index, specifying the task to be performed at that location. The map is then encoded as an image in the *STIF format*, described in Section IV-B.

Upon receiving the map, a node removes the image header and decompresses the task map. The node calculates the pixel in the image that corresponds to its physical location and retrieves the task index. At this point, the node compares the new task index to its current task index to determine if it needs updated code to perform the new task. Pseudo-code 1 shows the pseudo-code of the task map decoding algorithm.

Algorithm 1 Task Map Decoding Algorithm

```
if Receive a taskMap then
    mapVersion = getMapVersion(taskMap)
    if mapVersion < curVersion then
        curVersion = mapVersion
        [Rx,Ry] = getGISCoordinate(taskMap)
        [taskMap,imgWidth,imgHeight]=decodeMap(taskMap)
        pixelX = imgWidth*localX/Rx
        pixelY = imgHeight*localY/Ry
        taskIdx = taskMap(pixelX,pixelY)
    end if
end if
```

Figure 4 illustrates how Zoom works. To the left is the physical map of a city. An operator decides to measure noise pollution in the left area and to measure traffic speed in the right area. The operator defines the regions (e.g. by drawing

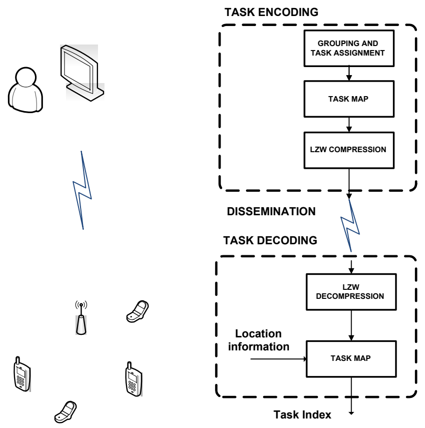


Fig. 3. Zoom Framework

on the map) and assigns the appropriate task indices, indicated by color values. The corresponding task map (on the right of the figure) is then encoded and disseminated to the network. A node upon receiving the map determines the task it must perform by checking the corresponding pixel value.

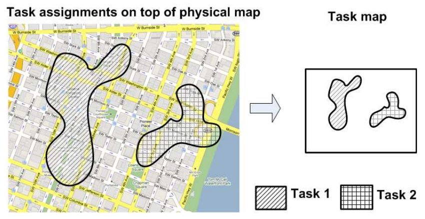


Fig. 4. Task Map Overlaid on Top of Physical Map

Suppose the two regions overlap. Then nodes in the overlapping area must perform both tasks. The overlapping area is assigned a new task index, which in turn includes both the two given tasks. The complexity of this operation is handled at the back end. Hence, the nodes themselves do not have to implement complex algorithms to interpret multiple tasks.

To avoid flooding the network with maps, we implemented a dissemination protocol with a *polite gossiping* mechanism [13] to reduce redundant message transmissions. Basically, each task map has a unique key and a version number that is incremented for each map update. A node upon receiving a new map will schedule a map rebroadcast. If it overhears other broadcasts of the same map and version, it will suppress its own rebroadcast and wait longer. This technique has been used in recent dissemination protocol [8].

B. STIF Format

A straightforward approach to encoding task and geographical information is to add attributes such as a task index to geographic information system (GIS) shapefiles [20], used widely in GIS applications, and disseminate this file into the network. This approach has two limitations. The shapefile has a large size as it was developed for a different purpose. A shapefile that lists all roads and suburbs in Portland is more than 7 MB whereas a very high resolution image representing the same information is only 800 KB. Moreover, it is difficult to represent the shapefile at multiple resolutions as roads are represented by a set of piecewise linear segments. It is unclear how fewer points could be used to represent roads at a lower resolution, without loss.

An image is good for representing arbitrary shapes. It can be resized to represent shapes at different resolutions quickly. We can also avoid unnecessary geometric computations to determine whether a node is inside or outside a region. The node can simply compare its location with the corresponding image pixel to determine its task index (pixel value). Image encoding and decoding functions are highly popular and have been built into almost every current hand held device. For these reasons, we choose images as the basic block for encoding task indices with group information.

There are several popular image file formats. Each format compresses images differently. Among them, Quadtree [7] and Graphic Interchange Format (GIF) [23] are suitable for Zoom. Quadtrees are most often used to partition a two dimensional space by recursively subdividing it into four quadrants. They encode the quadrants' locations together with the quadrants' values. However, as shown in table II, using the quadtree format still results in a large data size.

Format	Size (Byte)
Raw	65536
GIS Shapefile	68132
QuadTree	7735
Boundary	3192
STIF	1199

TABLE II
FILE SIZE FOR DIFFERENT FORMATS. STIF HAS THE SMALLEST SIZE.

In GIF [23], a color consists of three 8-bit channels — Red, Green, and Blue. GIF in general supports a maximum of 256 colors for its color index table. An image pixel is represented by one 8-bit value. The pixel value is the corresponding color in the color index table. Figure 5 depicts the core structure of a GIF image. The GIF representation is naturally suited to the purpose of our Zoom framework. Each pixel can describe a geographical region and the pixel value is the task index.

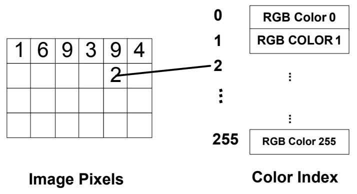


Fig. 5. GIF Image Format

The pixel values are compressed using the LZW compression algorithm [12]. LZW builds a dictionary for repeated patterns in the sequence of image pixel values - scanned from left to right and top to bottom - and encodes these patterns with a codeword that has a shorter length compared to the patterns. It must be noted that even without the dictionary, the image can be decoded successfully. Readers can refer to [12] for a complete description of GIF and LZW compression.

We have developed the Sensor Task Interchange Format (STIF) based on GIF. STIF represents a task map as a gray scale image and uses the same LZW compression technique as GIF. We, however, replace the GIF header with a simple header containing the map identification, the map version, the coordinates of the physical map, and the height and width of the image. Figure 6 depicts the STIF header. The id and version fields identify the map and the freshness of the map. The top left and bottom right coordinates scope the physical region to be reprogrammed. The image width and height indicate the size of the encoded image.

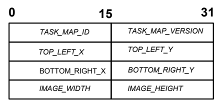


Fig. 6. STIF Header.

STIF has two drawbacks. The number of color indices is limited to 255 (8 bits/pixel). We believe that this is large enough for multiple concurrent tasks in a sensor network. We could increase the number of bits to represent a pixel, and consequently the number of task indices, but at the cost of compression efficiency. Moreover, decoding STIF images may require slightly higher memory (albeit smaller than the image size itself) compared to other image formats. By carefully tuning the LZW compression parameters, we can overcome the memory problem. Indeed, Sadler *et al.* [19] have developed an LZW variant for resource poor embedded devices.

Sometimes, a node may not have enough resources to decode a high resolution map representing a large geographical region. It also needs to know only the task indices in a small geographical region around itself. Instead of decoding the whole task map, we have developed three resource adaptation techniques, described in the next section, that can help Zoom conserve memory and bandwidth for such nodes.

C. Zoom Resource Adaptation

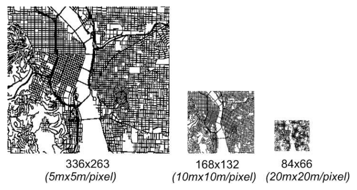


Fig. 7. Multi-resolution Encoding

1) Multi-Resolution Encoding: Our first resource adaptation technique is to encode the task map at different resolutions, allowing nodes to download only the appropriate resolution that they can handle. Figure 7 shows a map encoded at three different resolutions. A lower resolution leads to a smaller image, requiring less memory and computational power to decode. There is a trade-off between the resolution and the ability to define the task at a fine granularity. We analyze this trade-off in Section V-C1.

2) Selected Region of Interest Encoding: In Figure 8, a node needs to know the task indices of only a region large enough to cover its entire mobility (e.g. from home to work and back), rather than the task indices of all regions in the map. Instead of encoding the entire task map, we can selectively encode only a small region within the map. Hence, the node can obtain a region of interest (ROI) in the task map. In addition, a node may need high granularity task indices for a specific area such as a building to determine the appropriate task to perform when it is inside or outside the building. We can also selectively encode that region with a higher resolution. Hence, the node can obtain a higher resolution task map for the region of interest.

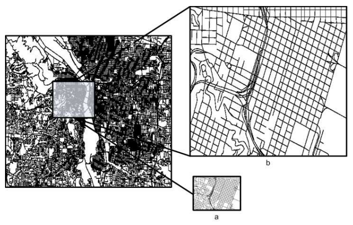


Fig. 8. Selected Region of Interest Encoding. A specific region can be encoded to reduce the image size to be transmitted. The region can also be encoded at a higher resolution to increase the identification accuracy.

Fortunately, GIF allows us to easily specify a specific region within the image to be encoded. After the header information, the data starts with a fixed image descriptor code followed by **TL** and **BR** (Figure 10) containing the top left and bottom right of the encoding image block. Using these two fields, we can specify the region of interest block. There is no change in decoding the map. However, this technique assumes that a node can interact with a base station.

3) Region of Interest Cropping: Upon receiving an encoded map, a node does not necessarily decode the whole map, either because it is interested in only the task index of its nearby region, or because it has limited resources and cannot decode the whole map. Zoom adapts a technique that allows the node to quickly crop out only a region that is relevant to itself. This technique was originally developed to support region cropping in video streaming applications [6].

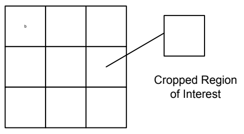


Fig. 9. Region of Interest Cropping. The task map is divided into blocks and encoded separately.

The main idea is to divide the task map into blocks and to encode each block independently (see Figure 9). The encoded blocks are appended to each other (Figure 10). Upon receiving the encoded map, a node can determine the corresponding region of interest based on its location, then search for the start of that block, and decodes only the found block. This technique does not conserve bandwidth but can help a node find its task index quickly with fewer resources.

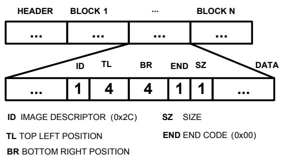


Fig. 10. GIF Format to Support ROI Cropping. Each image descriptor can be the start code for a new block.

In GIF, the image descriptor code (0x2C) is used to indicate the start of a block. All other fields remain the same. This way, we can use the GIF decoder with minimum changes, except to search the appropriate block to be decoded. However, dividing the map into smaller blocks reduces the LZW compression performance, increasing the total encoded map size. This is because the number of repeated patterns is reduced by limiting the data within blocks. We analyze the trade off between encoded map size and decoding time in Section V-C4.

D. Implementation

The Zoom framework has several components. We have implemented a task map encoder with support for Region of Interest (ROI) cropping in Matlab. We have implemented two variants of the task map decoder in C/C++; a basic task map decoder and a task map decoder with support for ROI cropping. The decoders are implemented in standard C. Hence they can be ported to different platforms using appropriate cross-compilers. The implementation details for the Intel(R) Core(TM)2 Duo chipset are listed in Table III. We have implemented a complete system with both the task map decoder and networking support in our simulators. [1]

Components	Program size	RAM
Task map decoder	14.5 KB	1.3 KB
Task map decoder with ROI cropping	15.3 KB	1.5 KB

TABLE III
IMPLEMENTATION DETAIL

V. EVALUATION

A. Goal and Metrics

Our evaluation goal is to answer the following questions:

1) Can the map-based approach in Zoom successfully represent geographical regions and tasks?
2) How is Zoom's performance in terms of encoded map size and update latency affected by the number of regions and number of nodes in the network?
3) Is Zoom better than previous approaches to macroprogramming?

[1]The preliminary version of Zoom is available for download at http://sys.cs.pdx.edu/home/projects/zoom.

To answer the first question, we consider how well STIF can encode road segments from a GIS file. Basically, we encode a geographical map using the STIF format and analyze the number of pixels that contain more than one road segment. Figure 11 depicts possible cases where a pixel may contain only one road segment (a), two road segments (b), three road segments (c), or four road segments (d). In the ideal case, a pixel should uniquely identify a road segment, containing no more than one road segment. However, as the map resolution is reduced, a pixel covers a larger geographical region and may contain more road segments. It is impossible to distinguish these road segments based on the pixel alone. In that case, STIF is unable to assign a distinct task to each road segment within a pixel. We refer to such a pixel as an *error pixel*. The definition of *error* also depends on the application. For example, in Figure 11(b), there is a clear error because the two roads do not intersect and are indistinguishable. Whereas the error pixel in Figure 11 (c) or (d), might be acceptable for some applications. In our evaluation, we consider (b), (c), and (d) as error pixels. We analyze the number of error pixels as a function of the map resolution.

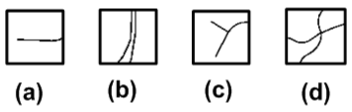


Fig. 11. Number of roads within a pixel. (a) a pixel can uniquely identify a road segment. (b, c, d) a pixel can not uniquely identify a road segment.

To answer the second question, we analyze (i) the update latency as a function of the number of nodes and map resolution in a realistic traffic monitoring application, (ii) the mean number of nodes with an incorrect task index as a function of map resolution, (iii) the size of the encoded task map as a function of the number of ROI blocks and (iv) the time to decode one specific block. In general, as the number of ROI blocks increases, the encoded map size increases but the decoding time decreases.

Answering the third question is somewhat tricky. It is not really possible to quantitatively compare Zoom to previous tasking approaches because as discussed in related work, each approach addresses a different problem and provides slightly different features. The work closest to Zoom is Logical Neighborhoods [16]. We compare the size of Zoom encoded maps to the size of Logical Neighborhood predicates for defining an equivalent spatial group of nodes.

B. Experimental Design

To investigate of the first question, we extract the GIS shapefile for Portland and its nearby suburbs from the latitude and longitude coordinates of (7597010.859, 645515.097) to (7696218.347, 711876.59). This covers a 600 sq.km area. The shapefile is available at the US Census Bureau website and contains several records. Each record contains one or more street segments. Each street segment is defined by a set of points with corresponding longitude and latitude coordinates.

We export this file into STIF files at different resolutions. Then we count the number of *error pixels*, as shown in Figure 11. These results are discussed in Section V-C1.

To investigate the second question, we use MobiReal [11], a realistic network simulator for mobile ad-hoc networks, to simulate a realistic traffic application. MobiReal is built on top of GTNets [18], a full-featured network simulator. MobiReal allows separation of behavior and network simulation. Therefore, we can specify realistic behavior models for cars or pedestrians and integrate them with the network simulator.

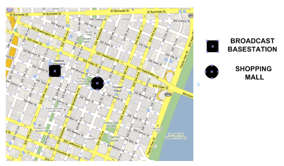


Fig. 12. Simulation Area. 1 km × 1 km area in downtown Portland, OR.

We simulate an *Advertisement* application in a 1 km × 1 km area in downtown Portland (Figure 12). The main components of the simulation are:

- *Advertisement Application*: We arbitrarily define regions in the map and assign different task indices for those regions. A base station (marked as a black circle in Figure 12) broadcasts the encoded task map to the network every 15 seconds. A car, upon receiving the map, decodes the map and updates its task index. The car also schedules map rebroadcasts. Together with the task map, the base station also broadcasts a sale advertisement for a nearby shopping mall (marked as a black square in Figure 12).
- *Mobility Behavior*: Cars are generated based on predefined road density that is close to the real density. A random entry point and a random destination are generated for each car at initialization. Cars travel to their destinations on the shortest path routes. However, upon receiving a sale advertisement, a car may add the mall address as an intermediate destination with a predefined probability (10% in our simulation). A car arriving at the mall stays at the mall for several minutes before leaving for the final destination. After arriving at the destination, the car is removed from simulation and a new car will be generated randomly. This mobility behavior allows us to simulate dynamic behavior inspired by a real scenario.
- *Networking*: Table IV shows the full network stack used in the simulation. Cars communicate with each other and with the base station using IEEE 802.11. The communication ranges are set to 100 meters for the cars and 300 meters for the base station. Both the cars and the base station use UDP as their transport protocol and IP as the network layer protocol. A car upon receiving the task map schedules periodic map rebroadcasts with an interval of 5 seconds. However, if it hears a broadcast of the same task within this interval, it suppresses its transmission and doubles the broadcast period interval. The maximum interval is set to 150 seconds.

Layer	Class	Description
Application	Advertisement	
Presentation	STIF format	Modification of GIF
Transport	L4Protocol	Wrapper class for UDP
Network	L3Protocol	IP V4
Routing	MyRoutingDSR	Dynamic source routing
Link (MAC)	L2Proto80211	802.11
Physical	DynamicWirelessLink	

TABLE IV
DESCRIPTION OF NETWORKING STACK USED IN SIMULATION.

We analyze the (i) update latency versus number of nodes (cars) in the network and (ii) the average number of nodes with incorrect task indices versus map resolution. Experimental results are shown in Sections V-C2 and V-C3. We also encode task maps with different number of blocks and analyze the map size and the time to decode a specific block. We also define regions arbitrarily to assign task indices and analyze the encoded map size versus the number tasks. Experimental results are shown in Sections V-C5 and V-C4.

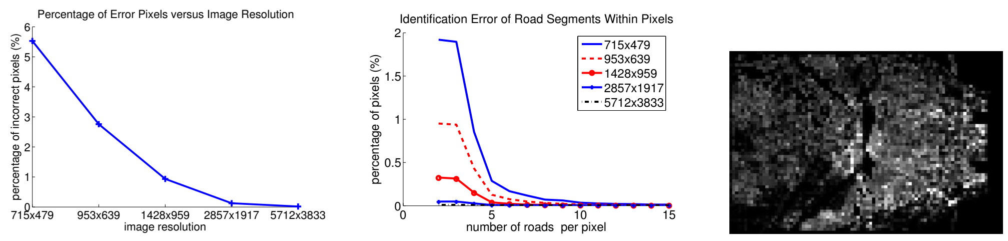
Finally, to investigate the third question, we define a number of regions randomly on a map and encode the map using Zoom. The number of regions are varied from 1 to 10. Using Logical Neighborhoods, we define the region boundaries and embed them in the predicate. Nodes with locations satisfying the predicate, *i.e.* their locations lie inside the regions' boundaries, are members of the corresponding groups. We compare the size of the Zoom STIF files to the size of Logical Neighborhood predicates to find out which approach requires fewer data transmissions. These experimental results are discussed in section V-C6.

C. Results and Analysis

1) Task Decoding Error: Figure 13(a) plots the percentage of error pixels (pixels containing more than one road segment) versus the map resolution. The number of error pixels decreases when the map resolution increases. With resolution 2857 × 1917, which is equivalent to a 10 m× 10 m square per pixel, the percentage of error pixels is almost zero. The black and white version of the map is only 321 KB in size. This is much smaller than the shapefile, which is 7MB. With resolution 715 × 479, which is equivalent to a 40 m × 40 m square per pixel, the percentage of error pixels is around 5.5% while the encoded map size is only 34.5 KB. Hence, the map-based approach in Zoom is suitable for representing geographical regions and tasks.

Figure 13(b) shows the distribution of error pixels. As expected, the higher the map resolution, the lower the error. Nevertheless, most error pixels contain 2 to 5 road segments.

Figure 13(c) plots the distribution of error pixels for a map of resolution 90 × 60. The whiter the color, the higher the number of roads collided within the pixel. Most high error pixels are distributed near the downtown and freeway intersection areas. This error distribution map is useful because we can increase the map resolution to decrease the identification error, when deploying a task map over a high error region.



(a) (b) (c)

Fig. 13. a) Total Error Pixels versus Resolution: With size of 321 KB at resolution 2857 × 1917, a STIF image can uniquely describe every road segment. b) Identification Error: Most error pixels contain 3 to 4 road segments. c) Distribution of Error Pixels: High error pixels (white color) are distributed near the downtown and freeway intersection areas.

2) Update Latency for Different Number of Nodes: Figure 14(a) plots the number of nodes with incorrect task indices versus time for different network sizes. The simulation scenario is an *Advertisement* in a 1 km × 1 km square in Portland, OR. The base station broadcasts the encoded task map at the 15th second. In the first 15 seconds, no node has the right task index. After the base station broadcasts the map, nodes update their task indices and rebroadcast the map. Hence, the error rate decreases quickly. However, due to nodes joining and leaving the network dynamically, we can not achieve a zero error rate. The error rate reaches a stable threshold after a while. Also, the higher the density, the lower the error.

3) Update Latency for Different Map Resolutions: Figure 14(b) shows the percentage of nodes with incorrect task indices versus time, with the task map encoded at different resolutions. The simulation scenario is 1 km × 1 km square in downtown Portland, Oregon. The task map is encoded at 191 × 155, 96 × 78, and 48 × 39 resulting in the sizes of 1.48 KB, 1.15 KB, and 0.975 KB respectively. The corresponding area per square are 5 m × 5 m, 10 m × 10 m, and 20 m × 20 m. The smaller the encoded map size, which correspond to a larger geographical area per pixel, the higher the error.

4) Trade-off Between Compression and Decoding Speed in ROI Cropping: Figure 14(c) plots the encoded map size of the same resolution versus number of blocks within the map. The original map size is 717 KB with resolution 1536 × 1536. The map is divided into several blocks, with each block encoded independently. The encoded map size decreases at first as the map is divided into 4 blocks. After that, the map size increases with the number of blocks. The peak at 4 blocks is because in the original map, the limited size dictionary of repeated patterns in LZW does not optimally capture the most frequent patterns over the entire map.

Figure 15(a) plots the average decoding time for a randomly chosen block. As we expected, the decoding time decreases quickly as the map is divided into more blocks. One should consider the trade-offs between the encoded map size (Figure 14(c)) and the decoding time desired (see Figure 15(a)) when applying Zoom to different hardware platforms.

5) Task Map Size versus Number of Regions: Figure 15(b) plots the encoded map size versus the number of regions. The map size grows in proportion to the number of regions. This is reasonable as the number of recurrent patterns in the map usually decreases when the number of distinct pixel values in the map increases.

6) Encoded Map Size versus Logical Neighborhood Predicate Size: Figure 15(c) plots both the size of the STIF maps and the size of the Logical Neighborhood predicates when encoding different numbers of regions. The regions are selected randomly and the number of regions are varied from 1 to 10. STIF maps always have smaller size compared to Logical Neighborhood predicates. This implies that Zoom potentially uses less bandwidth than Logical Neighborhood and sensors in Zoom use less memory than in Logical Neighborhoods.

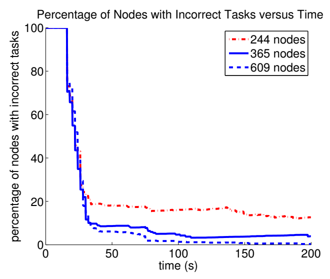
VI. Conclusion

We have presented Zoom, a multi-resolution tasking framework for crowdsourced, geo-spatial heterogeneous sensor networks. Zoom's innovation is to support heterogeneous devices by decoupling the task information (i.e. what task a sensor node must perform) from the task implementation (code). Zoom uses maps to represent task information and regions and encodes them in our proposed Sensor Tasking Interchange Format (STIF), making tasking intuitive for network operators. The use of maps also allows a mobile sensor to quickly obtain its task information without running complex geometric algorithms to determine whether it belongs to a region or not. We have also presented three resource adaptation techniques to reduce memory, bandwidth and CPU usage in Zoom.

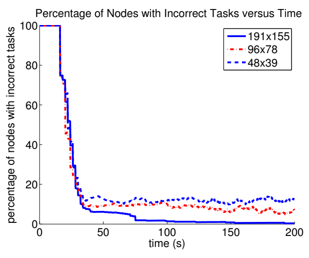
Our evaluation shows that Zoom is capable of tasking arbitrary groups of sensors in a large geographical network. With an encoded map of size only 34.5 KB, Zoom can task a region of 600 sq. km with only 2 % error. In addition, simulation of a realistic traffic application over an area of 1 sq. km with a task map of size 1.48 KB shows that more than 90 % of nodes are tasked correctly. Finally, for the same task information, Zoom's encoded map size is always 50% smaller than the predicate size in the state-of-the-art Logical Neighborhoods approach. To the best of our knowledge, this is the first work to propose an map based tasking framework for crowd-sourced geospatial sensing systems. We believe that Zoom's tasking capability is a step toward providing structure in increasingly unstructured mobile geo-spatial sensing systems.

References

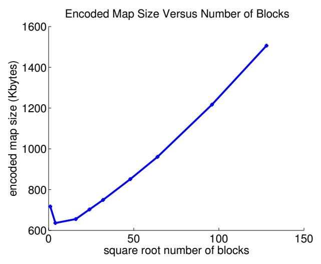
[1] T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. Stankovic,



(a)

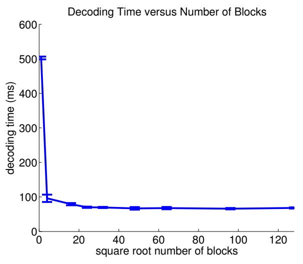


(b)

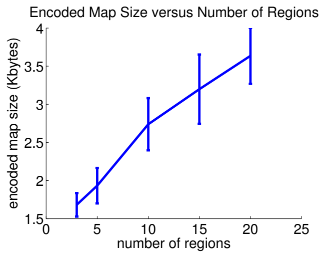


(c)

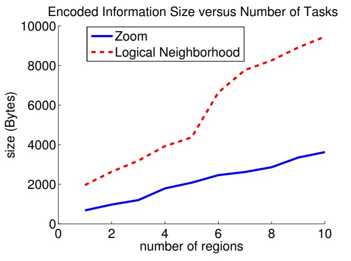
Fig. 14. a) Percentage of Nodes with an Incorrect Task Index versus Time: The higher the node density, the lower the error. b) Percentage of Nodes with Incorrect Tasks versus Time: The higher the map resolution, the lower the error. c) Encoded Map Size versus Number of Blocks.



(a)



(b)



(c)

Fig. 15. a) Decoding Time versus Number of Blocks. b) Encoded Map Size versus Number of Regions: Encoded map size increases proportionally with the number of regions. c) Encoded Information Size versus Number of Regions.

R. Stoleru, and A. Wood. Envirotrack: Towards an environmental computing paradigm for distributed sensor networks. In *Proceedings of ICDCS'04*, pages 582–589, Washington, DC, USA, 2004.

[2] S. Amin, S. Andrews, S. Apte, J. Arnold, J. Ban, M. Benko, R. M. Bayen, B. Chiou, C. Claudel, C. Claudel, T. Dodson, J. carlos Herrera, R. Herring, Q. Jacobson, T. Iwuchukwu, J. Lew, X. Litrico, L. Luddington, J. Margulici, A. Mortazavi, X. Pan, T. Rabbani, T. Racine, E. Sherlock-thomas, D. Sutter, and A. Tinka. Mobile century using gps mobile phones as traffic sensors: A field experiment. In *15th World Congress on Intelligent Transportation Systems 2008*, page 18, 2008.

[3] U. Bischoff. Rulecaster: A macroprogramming system for sensor networks. In *OOPSLA Workshop on Building Software for Sensor Networks*, 2006.

[4] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *Workshop on World-Sensor-Web (WSW06)*, pages 117–134, 2006.

[5] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson. People-centric urban sensing. In *Proceedings of the 2nd WICON 06*, page 18, New York, NY, USA, 2006. ACM.

[6] W. chi Feng, T. Dang, J. Kassebaum, and T. Bauman. Supporting region-of-interest cropping through constrained compression. In *Proceeding of ACM Multimedia 2008)*, pages 745–748, Vancouver, 2008.

[7] Y. ching Chang, B. kai Shyu, and J. shung Wang. Region-based fractal image compression with quadtree segmentation. In *in Proceedings ICASSP-97*, pages 3125–3128, 1997.

[8] T. Dang, N. Bulusu, W.-C. Feng, and S. Park. Dhv: A code consistency maintenance protocol for multi-hop wireless sensor networks. In *Proceedings of EWSN '09*, pages 327–342, 2009.

[9] D. Golovin, M. Faulkner, and A. Krause. Online distributed sensor selection. In *Proceedings of the 9th ACM/IEEE IPSN '10*, pages 220–231, New York, NY, USA, 2010. ACM.

[10] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of ACM Sensys 04*, pages 81–94, New York, NY, USA, 2004. ACM.

[11] K. Konishi, K. Maeda, K. Sato, A. Yamasaki, H. Yamaguchi, T. Higashino, and K. Yasumoto. Mobireal simulator evaluating manet applications in real environments. In *Proceedings of the MASCOTS 2005*, page 537, Atlanta, GA, September 2005.

[12] A. Lempel and J. Ziv. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(1):337–342, 1977.

[13] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of NSDI 04*, pages 2–2, Berkeley,CA, 2004.

[14] A. Liu, Y.-H. Oh, and P. Ning. Secure and dos-resistant code dissemination in wireless sensor networks using seluge. In *Proceedings of IPSN 08*, pages 561–562, Washington, DC, USA, 2008.

[15] S. Madden, M. J.Franklin, J. Hellerstein, and W. Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Transaction on Database System*, 30(1):122–173, March 2005.

[16] L. Mottola and G. P. Picco. Logical neighborhoods: A programming abstraction for wireless sensor networks. In *DCOSS*, pages 150–168, 2006.

[17] R. Newton, G. Morrisett, and M. Welsh. The regiment macroprogramming system. In *Proceedings of the 6th IPSN 07*, pages 489–498, New York, NY, USA, 2007. ACM.

[18] G. F. Riley. Simulation of large scale networks ii: large-scale network simulations with gtnets. In *Proceedings of the 35th WSC 03*, pages 676–684. Winter Simulation Conference, 2003.

[19] C. M. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *Proccedings of ACM Sensys 06)*, pages 265–279, Boulder, Colorado, Nov. 2006.

[20] Shapefile. Esri white paper. *ESRI Shapefile Technical Description*, 1(1):1–34, 1998.

[21] M. Welsh and G. Mainland. Programming sensor networks using abstract regions. In *Proceedings of NSDI 04*, pages 3–3, Berkeley, CA, USA, 2004.

[22] K. Whitehouse, C. Sharp, E. Brewer, and D. Culler. Hood: a neighborhood abstraction for sensor networks. In *Proceedings of the MobiSys 04*, pages 99–110, 2004.

[23] L. Wood. Graphics interchange format(sm) version 89a. *Graphics Interchange Format Programming Reference*, 1(1):1–34, 1990.

[24] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, 2002.