Anonymous Sensory Data Collection Approach for Mobile Participatory Sensing

Chih-Jye Wang, Wei-Shinn Ku

Department of Computer Science and Software Engineering
Auburn University, AL, USA 36849
{wangchj, weishinn}@auburn.edu

Abstract— **In participatory sensing, users with mobile devices participate in the collection of environmental information around them and submit the collected data to a central server for processing and analysis. Identity privacy of mobile users in this model is a major concern that hinders the adoption of this data collection model. In addition, bandwidth and computing capability are limited for mobile devices. Unfortunately, most previously proposed methods for user anonymity are not designed specifically for mobile environments and thus resource constraint is not the main focus of their solutions. In this paper, we propose One-way, an anonymous sensory data collection approach designed particularly for mobile participatory sensing environments. Our method utilizes peer-to-peer networks to facilitate anonymous data transmission to protect sender identity, but the actual data payload is not sent through peers. For data packets, we remove sender information and transmit through a direct path to the server. Since data are not replicated to multiple peers as in some previous solutions, our design consumes less resources. The experiments show that our approach utilizes less bandwidth and achieves higher scalability than existing techniques.**

I. Introduction

In participatory sensing [6], [14], users with mobile devices (e.g., cell phones with GPS chips) participate in the collection of environmental information around them and submit the collected data (image, audio, video, etc.) to a central server for process, analysis, and storage [11]. An example is the OpenStreetMap [7] project, which collects user contributed GIS data (such as GPS traces and POIs) and constructs an online map from mobile data, open for the public to use. Other examples of participatory sensing include cooperative collection of gas prices [5], monitoring of traffic conditions [3], and photo sharing through mobile applications such as Facebook and Instagram.

This model of "crowd" data collection has many advantages because the dynamic environmental data and information are quickly captured by mobile users and disseminated to other users or to central servers for analysis. Government censoring of information is also less effective in this model since there could be many mobile devices that are capturing the same event.

Despite the advantages, privacy of the users who contribute is a major issue in participatory sensing and crowd-sourcing. Users may not want to leak their locations with the contributed data lest their movement should be tracked [18], [20]. In the case of controversial photos, a user may not want to reveal his or her identity to the collection server (such as WikiLeaks[1]) since the user's identity may be saved on the server and later uncovered by government entities for prosecution. It is a fact that services on the web collect user identifications along with the data submitted by their users [19]. Even websites such as Wikipedia[2] and 4chan[3] that advertise to be anonymous log IP addresses of contributors. Although an IP address does not uniquely identify an individual, cellular data service providers are required by law to keep record of assignment of IP addresses to mobile users for a period of time[4]. With an IP address, a piece of information stored on a server (be it a photo, or a file) can be traced back to a user using the technique discussed in [17] thus compromises the user's privacy.

Many previous works [4], [10], [16] have addressed the ways to guarantee that a service provider will not gather users' identification on the Internet. Unfortunately, most previously proposed solutions are designed for the older computing model of desktop computing where network bandwidth and computing power are not major concerns. In mobile environments, both bandwidth and computational capability are limited partially due to the size of devices and battery power constraints.

In this paper, we propose an anonymous data collection model for participatory sensing with the constraints of mobile environments in mind. The goal is to design a solution that is simple so that it does not consume too much resource for mobile devices (resources being bandwidth and computational power), and at the same time provides adequate protection for privacy. We want to keep the identity of data owners (senders) private to various parties in a computer network, especially to the data collection server (receiver) since without identity information associated with submitted data on the server, linking to a data originator is difficult. Although we mainly focus on keeping the sender anonymous to the server, we believe our solution has enough protection from other parties in a network. Our mechanism is to remove the source information from data packets. We call this One-Way Protocol since the source information is completely removed, and the server is unable to reply with any kind of acknowledgement

[1]http://wikileaks.org/
[2]http://www.wikipedia.com
[3]http://www.4chan.org/faq#what4chan
[4]http://uscode.house.gov/download/pls/18C121.txt

back to the client. The protocol will be implemented on both the server and the client to achieve the anonymous effect.

The rest of this paper is organized as follows. In section II, we survey previous works for anonymity on data communication. For each work, we identify strengths and weaknesses relative to our design goals. We introduce our anonymous One-Way data collection approach in Section III. We discuss how data payload is transmitted and how uploaded data is verified although the server does not know the identity of the sender. Section IV presents the implementation issues and Section V discusses a few attack models. The experimental validation of our design is presented in Section VI. Section VII concludes this paper with a discussion of future work.

II. Related Work

One of the earliest work on anonymous communication is on untraceable electronic mail by Chaum [1]. The work proposes a way of sending an electronic mail to a receiver in which the sender remains anonymous using one or a series of proxy computers called *mix*. Chaum's solution relies heavily on public key cryptography in a way that before forwarding a mail to a mix, the message and the receiver's address is encrypted using the mix's public key. Upon receiving an electronic message, the mix extracts the information and forwards the mail to the intended receiver. In the case of multiple mixes, the sender performs one public key cryptography for each mix before sending a mail to the first mix. The mix then forwards it to the next mix and so on until the mail reaches the receiver. One of the drawbacks of the solution proposed in [1] is scalability since the senders depend on fixed proxies. If there are no proxies, the solution will not work. Also, multiple public key cryptography is expensive for mobile devices. Multiple public key encryptions are acceptable for small files, but for large files that need to be broken down and sent with multiple segments, multiple encryptions may not be acceptable.

Another well known mechanism of sender anonymity is the onion routing [4]. The onion routing follows the similar approach proposed by Chaum of using multiple proxies and multiple encryptions. Before sending a file, the sender selects a number of proxies called onion routers, which mitigates malicious node collaboration attacks. After selecting the routers, the sender encrypts the data with the key of the first router, then encrypts the result of that with the key of the second router, and does the same for all the other routers. Onion routing is robust in terms of security and can be used in many applications, but it has the same drawback as in Chaum's approach that it is computationally expensive because of the multiple encryptions. Our solution does not impose multiple encryptions to applications, but leaves that choice to the developer of applications.

Freenet, a distributed anonymous storage system, was proposed in [2]. The goal of the system is to utilize the peers in the network for storage and at the same time achieve anonymity of the authorship of the files. Freenet employs file keys obtained by public key cryptography and hashing to achieve efficient file storage and searching. In order to achieve author anonymity, a node on the Freenet sends a file to a peer, and the peer forwards the file to another peer and so on. Although the system pertains to file storage, the technique of utilizing peers to hide the identity of the originator of a file is useful in our scenario and we take a similar approach with data transmission verification of our design.

The Crowds technique [16] utilizes peer-to-peer networks to blend the sender into a large number of users (a crowd) so that the receiver does not know the true identity of the sender. To send a file, the sender joins a close-by crowd and sends the file to one of her peers in the crowd called jondo (a short name for John Doe). When a peer receives a send request, she decides whether to forward the data to another peer based on the variable p_f that determines the probability of forwarding the packet.

The closest work to ours is the research on privacy assurance for mobile users in sensing networks by Hu and Shahabi [8]. The assumption of the work is that data collection service providers often log the identities of users with the file they submit. The disclosure of this sensitive information leads to the compromising of user privacy. The solution is to leverage social network to forward a file to multiple peers before sending it to the server. Large consumption of network bandwidth is a significant disadvantage in this approach since a file is sent to multiple users, and this concern is exacerbated in wireless mobile environments where bandwidth is limited. In our approach, we do not forward the sensor collected data to multiple peers; we only forward the control messages of the data transmission and save a tremendous amount of bandwidth.

III. Anonymous Sensory Data Collection

A. Overview

In this section, we describe our anonymous One-Way protocol, which completely removes the ownership and source information from the sensory data transmitted to the data collection server. Figure 1 illustrates the concept of the theoretical anonymous One-Way data collection model. One-Way

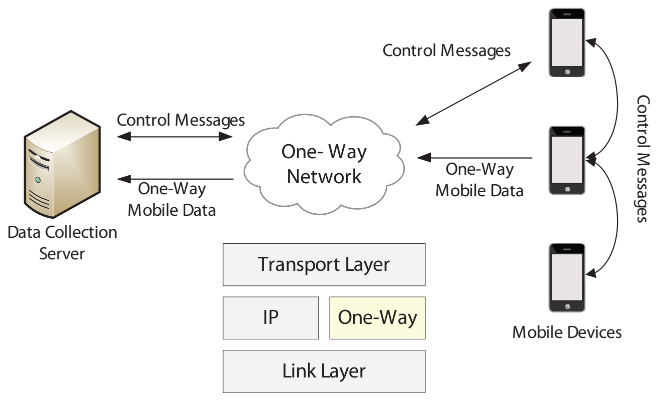

Fig. 1. Anonymous One-Way data collection model.

employs a packet routing protocol similar to the IP protocol. Data are transmitted in packets. Each packet contains the address of the destination host so that the packet can be correctly routed. One major difference between the One-Way protocol and other routing protocols is that each packet does not include the routing address of the sender (thus called One-Way). The source address is intentionally removed so that the receiver of the data does not know the identity of the sender. In other words, contractually, the client does not include the source address and the server will have no access to the source information, thus keeping the client's identity private. Our protocol is analogous to sending a parcel through a postal service without the sender's identity and address. An example is sending a hint to the police pertaining to a crime without revealing the sender's identity.

Unlike many other anonymous routing protocols, our protocol does not involve other peers or mobile devices to transmit the data payload. This is particularly helpful in mobile environments in which bandwidth is limited. Some privacy aware protocols first transmit the data payload to several mobile peers several times to mask the identity of the sender before one of the random peers transmits the data to the server. This incurs tremendous bandwidth strain on the peers.

One limitation of a protocol without sender address is that the sender is unable to get an acknowledgement from the receiver whether a packet has been received (or lost in transmission). To solve this problem, One-Way utilizes a peer-to-peer approach to verify correct transmission of a packet, much like the way Crowds [15] employs to send data to a server.

For validation of successful transmission, we construct a peer-to-peer network containing h nodes, where h denotes the total number of hop counts for control messages to pass though. A peer node is denoted by n_i, meaning the i-th hop along the route. We use n_0 to denote the originator of the data, and n_h the last peer that forwards the data to the data collection server. For control messages travel in the direction from sender to receiver, a node n_i forwards the message to its successor, such that $n_i \rightarrow n_{i+1}$. Similarly, for control messages travel from the receiver back to the sender, a node forwards the messages to its predecessor, such that $n_i \rightarrow n_{i-1}$. Table I summarizes the notations used in this paper.

Opponents may think that since the protocol resides on the network layer (layer 3), the source address could still be contained in the lower layer, e.g., the data link layer (layer 2). We argue that the source link address is different for each hop on the route, and a route contains at least 10 hops; therefore the probability of tracing the link layer packet back to the original host is very low.

B. Anonymous Connection

Before transmitting data to a data collection server (receiver), the sender makes an anonymous connection request to the server. The purpose of the connection request is to establish a peer-based route on which acknowledgements of data transmission can be sent.

The sender initiates a connection by constructing an anonymous connection request message, c. A request message contains the connection identifer I_s, routing address (or identification) A_r of the receiver, and a hop count h. The structure of the request message is defined in Equation 1

$$c = \{I_s, A_r, h\}, h \geq 10 \tag{1}$$

In the connection request packet, h is a greater than ten random natural number denoting the number of peers the request should be forwarded to before the request is sent to the server. The connection identifier I_s is randomly generated, which only lasts for the transmission of a sensory file (e.g., a photo) and does not directly identify a host.

After the sender has constructed the connection request message, she sends the message to one of her peers running the protocol. When the peer receives the request, it decreases the hop count h by 1. If the hop count reaches 0, then the peer sends the request to the server; else, the peer forwards the request to another one of its peers. Note that a peer (node) does not have to be a pure peer, but a node in the system whose job is to facilitate communications for user nodes. Each peer in the route keeps track of the peer before and after for each request in order to maintain the route information. If a peer declines a c message, the peer finds another peer. At the end of the connection phase, the peer-to-peer route should be constructed such that

$$sender(n_0) \rightarrow n_1 \rightarrow ... \rightarrow n_h \rightarrow receiver \tag{2}$$

When the server receives the connection request, it sends back an acceptance message a to the node (n_h) from which it receives the connection request. The acceptance message contains a connection token I_r, as well as the request identifier from the sender, I_s. The format of the acceptance message is defined in Equation 3

$$a = \{I_r, I_s\} \tag{3}$$

The node that receives the acceptance message from the receiver forwards the message in reverse to the peer who forwarded the request message. Each peer in the route keeps the route information based on I_s and the acceptance messages

TABLE I
SUMMARY OF NOTATIONS.

Notation	Description
n_i	A peer node
h	Total hop count
I_s	Connection identifier of sender
I_r	Connection identifier of receiver
A_i	Address of the ith node
A_r	Address of the receiver
D	Data payload
s_i	The ith sequence number
c	Connection control message
a	Acceptance control message
p	Data payload packet
k	Acknowledgement control message

travels in reverse on the same route until the message reaches the original sender.

C. Payload Transmission

Payload is the actual data transmitted to the server. In participatory sensing, this is the data collected by mobile devices from the environment and includes sensor data, photos, videos, etc. Payload transmission does not involve peers since peer-forwarding in wireless mobile environments is expensive; instead, payloads are transmitted through the One-Way network, in which the packets only contain the receiver address and identification.

The payload transmission begins when the sender receives the acceptance message from the server. In addition to the payload D, a payload packet, denoted with p, also contains the address of of the receiver A_r, the connection identifier I_r, and a sequence number s_i. The format is defined in Equation 4.

$$p = \{A_r, I_r, s_i, D\} \tag{4}$$

I_r is the identifier issued by the server received from the connection acceptance message. The purpose of this identifier is for the server to identify a particular data transmission. For example, the server could be receiving several data transmissions simultaneously from different clients each transmits a file. The server assigns an identifier I_r to each transmission to differentiate the packets for different transmissions. This field is analogous to the port numbers in the transport layer such as TCP and UDP to multiplex packets for different connections and applications.

The sequence number s_i in a payload packet identifies individual packets for a transmission. The sequence numbers provide the order of the packets and allow the receiver to identify if a packet is lost in the network, or if packets arrived out of order.

The purpose of the connection establishment step before payload transmission is to establish a channel in which the mobile client can verify that a packet has been successfully received by the server. Techniques of reliable data transfer in unreliable channels are out of the scope of this paper. Reliable data transfer mechanisms employed in TCP [13] and theoretical techniques such as GO-Back-N and Select Repeat [9] can be employed with our design. In addition, when the server sends the status of a packet back to the client, it sends the status message through the peer-to-peer channel established during connection establishment. In other words, after receiving a payload packet p, the server sends an acknowledgement message, k, to the node from which it received the connection request (also the node to which it sends the connect acceptance message). Since all the nodes on the anonymous route memorize the previous node based on the connection identifiers, I_s and I_r, each node forwards k to its previous peer until the message reaches the originator of p.

D. Node Disconnection and Failure

When a node can no longer service a route for a connection due to resource constraints or node crash, the disconnecting node (denoted by n_d) should notify its neighboring nodes of such event so that the interruption of a connection and of delivery of control messages can be minimized. In this section, we focus on disconnecting a single connection (as in the case of resource constraint). If a node wishes to disconnect all connections it supports (as in the case of node shutdown), n_d performs a disconnection process discussed in this section for each connection.

There are two main scenarios for n_d: the next node is an intermediate node, and n_d is the last node in the route ($d = h$). For the first case in which n_d is an intermediate node and the next node is also an intermediate node, n_d should simply bridge the gap and withdraw from the route. In this case, n_d sends the node before, n_{d-1}, and the node after ,n_{d+1}, a disconnection message containing the identity (the address) of the two nodes. Once the connection between the two nodes has been established, n_d can remove the route information for the connection from its routing table.

In the second case, n_d is the last node of the route and communicates directly with the server. In this case, we do not want just bridge the two nodes n_{d-1} and n_{d+1} since n_{d-1} could be the originator of the data. In order to establish a new route for the connection, n_d sends a disconnection message to the server telling the server that a new control message route will be constructed for this connection. n_d also sends a route discovery message to n_{d-1} telling the node to construct a new route to the server. n_{d-1} follows the same steps described in section III-B to establish a new connection to the server. The disconnection algorithm is detailed in Algorithm 1.

Algorithm 1 disconnect(n_d)

```
1.  if $A_d$ == $A_h$ then
2.      $m_1$ = DisconnectMessage($I_s$, $I_r$)
3.      $m_2$ = RediscoveryMessage($I_s$, $I_r$)
4.      SendToNode($A_r$, $m_1$)
5.      SendToNode($A_{d-1}$, $m_2$)
6.  else
7.      $m_1$ = BridgeMessage($Forward$, $A_{d+1}$, $I_s$, $I_r$)
8.      $m_2$ = BridgeMessage($Backward$, $A_{d-1}$, $I_s$, $I_r$)
9.      SendToNode($A_{d-1}$, $m_1$)
10.     SendToNode($A_{d+1}$, $m_2$)
11. end if
```

In the event of an unforeseen failure of a node, such as a power or network failure, the disconnected node n_d is unable to notify its neighbors on a route before disconnecting. When this happens the first thing that the sender will notice is that she no longer receives acknowledgement messages of received packets from the server. To test for node failure in the path, the sender can send a route test message into the route. When a node n_m receives the test message, it sends an acknowledgement back to its predecessor n_{m-1} on the route and forwards the test message to its successor n_{m+1} on the route. If an intermediate node on the route n_f cannot be contacted or does not acknowledge the test message, then its predecessor n_{f-1} sends a failure notice backward through the route to notify the sender that a failure has been detected. n_{f-1} uses the route discovery algorithm to discover a new route from n_{f-1} to the server and notify the sender that the new route has been successfully established. In the case that there is no failure detected but the sender still does not receive acknowledgements from the server, the sender can select to terminate the data transmission and peer connection.

E. Connection Tear-Down

After the entire file has been sent to and received by the server, the sender can choose to tear down the connection so that the nodes can free the resources used for the connection. To terminate a connection, the sender injects the termination message identified by I_s and I_r into the connection. Upon receiving the termination message, a node forwards the message to its successor and frees any resource associated with the connection. Eventually the termination message reaches the server. The server frees the resources and stores the received file.

IV. Implementation Issues

In this section, we discuss the issue of implementing the protocol proposed in the previous section. One issue is that the Internet currently does not support a protocol without sender information. We propose two ways of implementing our One-Way protocol: using trusted gateways, and on top of UDP transport layer protocol [12].

A. Trusted Gateway

In the trusted gateway method, we propose to implement our protocol in hardware with a wireless access point that understands our One-Way protocol. In the method, the network topology is divided into two parts: a private network that understands the One-Way protocol and the rest of the world that only understand the IP protocol as shown in Figure 2. The gateway acts as a normal wireless access point and multiplexes traffic for the clients inside the private network.

Since the control messages do not have the issue of concealing the sender routing address, the transmission validation route is constructed on the normal IP network. When a client intends to send a file to a data collection server, she finds a peer on the traditional IP network that runs our protocol. Afterwards, the client constructs a connection request message and sends the message to the peer. The first peer forwards the request to a second peer and so on until the hop count of the request and the request is sent to the server.

When sending payload, the sender sends payload packets through the trusted gateway. Upon receiving a payload packet p, the gateway performs a tunneling operation and transforms p into a standard packet including all necessary information for the header in the packet by using its own identity, then forwards the packet into the IP network. Since the gateway utilizes its own IP address as the address of the packets, it shields the identity of the mobile clients. One reason for this tunneling-based approach is that some Internet service providers (ISP) block packets without valid source host addresses. Figure 2 also depicts the networking stack configuration of this approach.

Opponents of this approach may think that attackers can still trace the mobile clients to a specific subnet where the gateway device resides. However, with a number of access points in one geographic region, we argue that since most of the clients are highly mobile, the identity of the mobile devices can be protected with high confidence.

B. Over User Datagram Protocol

If a subnet allows packets with anonymous sender IP addresses (such as the broadcast address) to be forwarded through the network, then we can build our protocol on top of existing network protocols. In this approach, we completely do away with trusted access points by building the One-Way protocol as an application layer protocol on top of the User Datagram Protocol (UDP) and employ existing network hardware and infrastructure. This method is illustrated in Figure 3.

The steps for making a connection request are the same for the trusted gateway approach, but payload transmission is different. During payload transmission, instead of letting a gateway use its identity for forwarding the packets, the sender uses a special address (e.g., the IP broadcast address) in the sender address field of the packets. This denotes that the sender is anonymous and the server should treat the data transmission by employing the anonymous data collection model.

Note that the reason we use UDP is that TCP will not work in our approach because it is a connection oriented protocol. With an arbitrary IP address field, TCP will be unable to establish a connection with the handshaking mechanism.

V. Attack Models

In this section we consider several attack models and discuss how our design offers protection against these attacks.

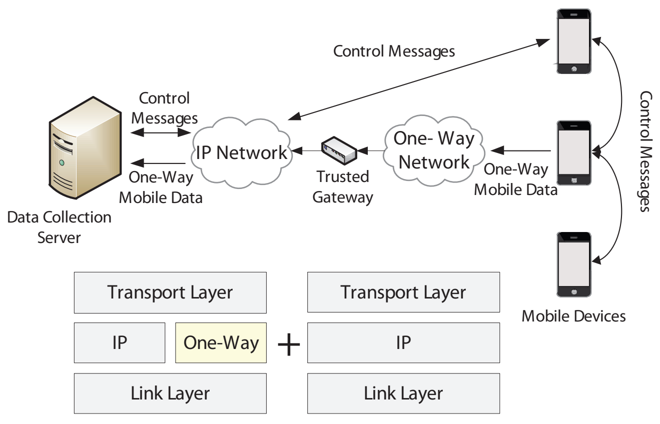

Fig. 2. Trusted Gateway.

A. Server Attack

The first attack model is server attack. In this attack, the server cannot be trusted or can be a source of identity of disclosure. Most services log the identity of users with the data submitted through their services. If the server is malicious, or mandated by some authority to disclose information, the privacy of users is compromised.

First, we analyze the payload transmission part of our design against server attack. We assume that the server can see the packets sent to it, but does not see the packets in the network before they reach it. In other words, the server does not control the Internet and cannot perform traffic analysis on the Internet. Under this assumption, in our One-Way protocol the sender has the anonymity level of beyond suspicion since a server cannot infer any identity information from the data packets it receives. The only linking information is the identity of the network device (e.g., link layer switch) at the last hop of the route. We assume the service does not control the Internet, therefore it cannot link this information back to the sender.

Next, we analyze our validation protocol against server attack. Keep in mind that our validation mechanism utilizes peer network to transmit acknowledgements and control messages. Since the server has to execute our anonymous validation protocol, it is considered one of the network peers and therefore it can find out the users in the peer network, but the server does not know the original hop count h of the control packets. In addition, any peer can potentially be connected with any other peers. Consequently, the sender is also beyond suspicion in this case.

B. Eavesdropper and Traffic Analysis

In eavesdropper attack, we assume that the data transmission route is divided into segments each controlled by a network administrator. An attacker is able to listen to the traffic of a segment, but cannot eavesdrop on all the network segments of the route. In other words, the attacker does not have the full view of the entire route, but only a segment.

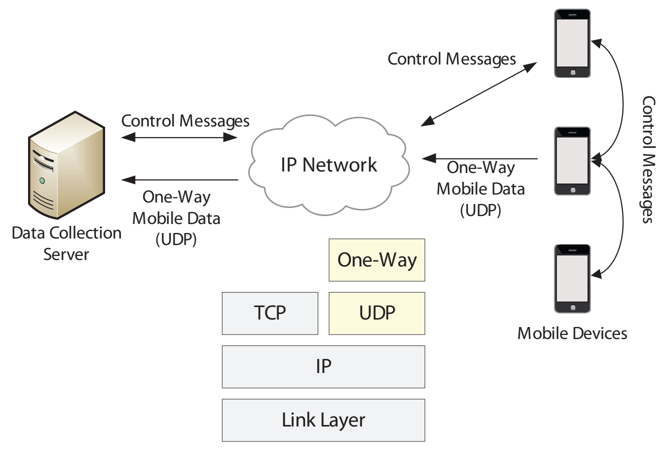

Fig. 3. One-Way over UDP.

There are two scenarios. The first scenario is that the sender of the data does not reside on the network on which the attacker is eavesdropping and the the level of anonymity for the sender is beyond suspicion. Although the attacker is able to listen to the traffic to her network, she is unable to see who the sender is in other network segments.

The second scenario is that the sender and the attacker are on the same network. In this case the attacker is able to analyze the traffic of any node on the network. In this scenario, our design is unable to protect the identity of the sender and the level of anonymity is exposed. The way that the attack could get the sender's identity is to compare the incoming and outgoing traffic for every node in the network. For the nodes that have outgoing traffic for a specific content without corresponding incoming traffic is suspected of data originator. Encryption of data on top of our design can mitigate this attack.

VI. Experimental Evaluation

In this section, we evaluate the performance of our anonymous sensory data collection approach. We implement the proposed solution and related experimental components in C++. All simulation results were recorded after the system model reached a steady state.

We study the transmission overhead of our design and compare the performance with anonymous communication protocols discussed in Section II. Transmission overhead is the additional data (in bytes) incurred by a protocol to facilitate transmission of the original data. Transmission overheads include packet headers and data redundancies. For our evaluation, we model our design on top of UDP. We assume that the overhead of our design includes the headers of UDP datagrams. Given D as the size (in bytes) of the original data, H as the size of protocol headers, and R as data redundancy, the percentage of overhead P is calculated as Equation 5

$$P = \frac{H + R}{D} \tag{5}$$

The techniques which we compare our solution with are Tor [4], Crowds [15], and HP^3 [8]. For Tor, we model the protocol overhead according to the description provided in [4]. Tor incurs one layer of header overhead (containing circuit information) for each onion router on the route and we define this overhead H_{tor} = 14 bytes. For Crowds, there are no implementation details specified in the study; therefore, we take a modest estimate of 4 bytes of header overhead (for storing the forwarding probability p_f number). Similarly, there is no implementation detail for HP^3, but the protocol employs a redundancy of β for error correction. We assume $\beta = 20\%$, which is the redundancy used by the authors. Furthermore, we include header overhead of TCP and UDP in our measurement. In addition, we assume the payload size of each packet to be 500 bytes. For payload size larger than 500 bytes, the payload is divided into multiple packets. Simulation parameters are summarized in Table II.

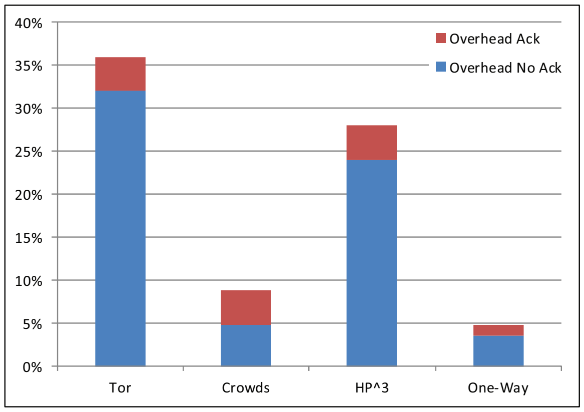

Fig. 4. Sender overhead percentage of 10 peer nodes.

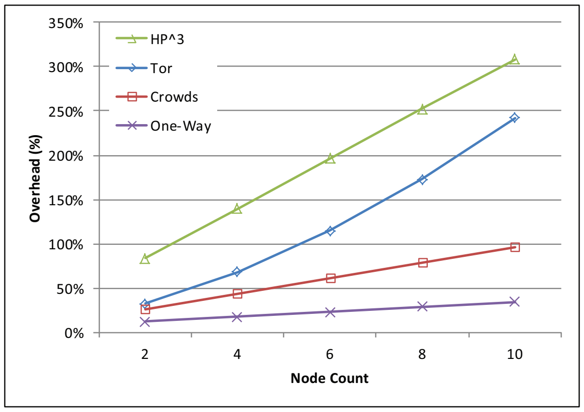

Fig. 5. Peer overhead percentage without data replication.

Figure 4 shows our evaluation result for sender overhead with 10 peer nodes. In the figure, the overhead without acknowledgement is the overhead of sending the data to the server and the overhead with acknowledgement is the overhead of receiving acknowledgements from the server, in addition to sending the data. The results show that our design incurs the lowest overhead of 3.6% without acknowledgements and 4.8% with acknowledgements. Additionally, Tor has the highest overhead of more than 30% due to the layering overhead of each node.

TABLE II
SUMMARY OF SIMULATION PARAMETERS.

Parameter	Size (bytes)
One-Way data header	10
One-Way acknowledgement	6
Tor header per layer	14
Crowds header	4
HP^3 redundancy (β)	$0.2 \times D$
TCP header overhead	20
UDP header overhead	8

Figure 5 illustrates the overhead of all the peers involved in data transmission. The overhead percentage is calculated as the sum of all transmissions of all involved peers over the size of the original data. The overhead does not take into account the replication of data. HP^3 has the highest overhead due to the 20% redundancy for error correction. Our mechanism has the best performance with 9% for 2 peers and 31% for 10 peers.

Figure 6 complements the previous figure by demonstrating overhead percentage of all peers and taking into account of the copying of D to involved peers. Unlike the previous simulation result, the measurements in this figure take into account the replication of data to the peer nodes. As shown in the figure, all protocols (except for ours) have a similar performance trend and HP^3 has the steepest growth due to the data redundancy. The result illustrates that the performance of our solution is not affected by the replication of data since our design does not replicate the data through peers.

VII. CONCLUSIONS

In this paper we propose One-Way, which is an anonymous sensory data collection technique to protect the identity privacy of mobile users in participatory sensing environments. Bandwidth and computational capability constraints of mobile devices are important considerations for our design. Our approach utilizes peer-to-peer networks to construct an anonymous route on which control messages (such as acknowledgements) can be sent to facilitate data transmission. On the peer route, each node keeps track of its predecessor and successor, but does not know the length of the route, thus keeping the identity of the sender anonymous. In order to conserve bandwidth in mobile environments, the actual data payload is not sent through peers, but instead sent through a direct path using the One-Way protocol. In the One-Way protocol, payload messages are sent to the server without the sender's identity (or address). When the server receives payload messages, it sends acknowledgements back to the sender through the anonymous peer-to-peer route.

We present two implementations for our protocol: using trusted gateway and on top of UDP transport layer protocol. Trusted gateway is useful when the ISP has a stringent policy of only accepting packets with a correct sender address. In the

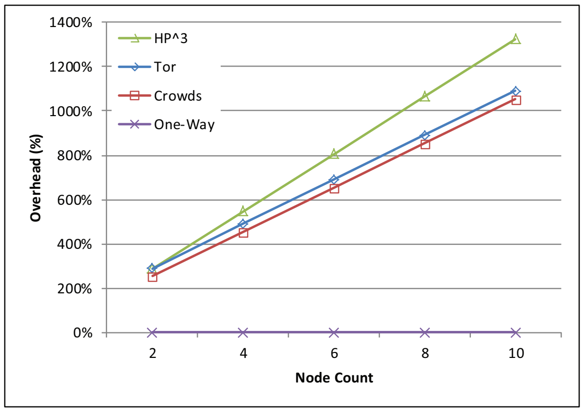

Fig. 6. Peer overhead percentage with data replication.

case of implementing on top of UDP, we utilize the broadcast address to replace the sender's address to signify that the packet should be treated as anonymous.

Our performance evaluations show that our solution performs well when we compare it with other privacy protection mechanisms in terms of resource utilizations. Our approach incurs significantly less transmission overhead (in terms of data amount) due to the fact that we only utilize peer network for small portion of the transmission. Transmission latency is also improved over other protocols. For future work, we plan to perform further analysis of our design, such as scalability, practical implementation issues, and adaptability by ISPs. We also plan to further investigate security issues and study more attack models for our design.

Acknowledgment

This research has been funded in part by the National Science Foundation (NSF) Grants CNS-0831502 (CT) and CNS-0855251 (CRI).

References

[1] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[2] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, 2000.

[3] Hugo Conceição, Michel Ferreira, and João Barros. On the Urban Connectivity of Vehicular Sensor Networks. In *DCOSS*, pages 112–125, 2008.

[4] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, pages 303–320, 2004.

[5] Yifei Dong, Salil S. Kanhere, Chun Tung Chou, and Nirupama Bulusu. Automatic Collection of Fuel Prices from a Network of Mobile Cameras. In *DCOSS*, pages 140–156, 2008.

[6] Prabal Dutta, Paul M. Aoki, Neil Kumar, Alan M. Mainwaring, Chris Myers, Wesley Willett, and Allison Woodruff. Common sense: participatory urban sensing using a network of handheld air quality monitors. In *SenSys*, pages 349–350, 2009.

[7] Mordechai (Muki) Haklay and Patrick Weber. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[8] Ling Hu and Cyrus Shahabi. Privacy assurance in mobile sensing networks: Go beyond trusted servers. In *PerCom Workshops*, pages 613–619, 2010.

[9] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. Addison-Wesley, 5th edition, 2009.

[10] Brian Neil Levine and Clay Shields. Hordes: a Multicast-Based Protocol for Anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.

[11] Min Mun, Sasank Reddy, Katie Shilton, Nathan Yau, Jeff Burke, Deborah Estrin, Mark H. Hansen, Eric Howard, Ruth West, and Péter Boda. PEIR, the personal environmental impact report, as a platform for participatory sensing systems research. In *MobiSys*, pages 55–68, 2009.

[12] Jon Postel. RFC 768: User Datagram Protocol, 1980.

[13] Jon Postel. RFC 793: Transmission Control Protocol, 1981.

[14] Sasank Reddy, Jeff Burke, Deborah Estrin, Mark H. Hansen, and Mani B. Srivastava. A framework for data quality and feedback in participatory sensing. In *SenSys*, pages 417–418, 2007.

[15] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Trans. Inf. Syst. Secur.*, 1(1):66–92, 1998.

[16] Michael K. Reiter and Aviel D. Rubin. Anonymous Web Transactions with Crowds. *Commun. ACM*, 42(2):32–38, 1999.

[17] Latanya Sweeney. k-Anonymity: A Model for Protecting Privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[18] Ting Wang and Ling Liu. Privacy-aware mobile services over road networks. *PVLDB*, 2(1):1042–1053, 2009.

[19] Wikipedia. Wikipedia Privacy Policy. Modified: 26/08/2011.

[20] Man Lung Yiu, Christian S. Jensen, Xuegang Huang, and Hua Lu. SpaceTwist: Managing the Trade-Offs Among Location Privacy, Query Performance, and Query Accuracy in Mobile Services. In *ICDE*, pages 366–375, 2008.