S2S: a peer to peer protocol for participative sensing

ABSTRACT
Participatory sensing represents a promising perspective in the worldwide diffusion of smarter and smarter mobile devices. Availability and shareability of gathered data from a potentially everywhere and every-time community of users represent a totally novel opportunity in a variety of fields such as security, public health, meteorology, sociology, etc. Challenges and imperatives to design and implement consistent systems for participatory sensing must generally address to precision, resources-savings, credibility, as well as to users’ privacy and non-invasive impacts. This paper proposes s2s, a p2p protocol to increase precision and effectiveness, as well as to reduce resources consuming on NoiseHound, a system which has been designed and implemented for participative sensing of noise pollution by means of common smartphones. S2s has been conceived with the aim to obtain an optimal scheme to lookup and query neighbors’ information for improving quality of sensing and reducing data exchanges through the Net.

Categories and Subject Descriptors

General Terms

Keywords
Participatory sensing, distributed systems, network protocols, simulation, wireless networks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SAC’12, March 25-29, 2012, Riva del Garda, Italy.
Copyright 2012 ACM 978-1-4503-0857-1/12/03...$10.00.

1. INTRODUCTION
Besides their estimated worldwide amount, which is over 5 billions thus far [1], mobile devices have shown an extraordinary penetration rate as they evolved towards sophisticated tools to communicate, plan, locate, inform, play with, entertain. Percentages of smartphones, in comparison with the total amount of handsets has been forecast to grow up to 17% worldwide by 2014 (from 9% in 2009), with values close to 50% in north America and western Europe (from 25-30% in 2009) [2]. General purpose processing capabilities of these devices have further increased the opportunities of transform, enhance and mash up data coming from cameras, compasses, accelerometer and GPS receivers they are equipped with. 3G and Wi-fi interfaces opened to smartphones the doors of the Net, by definitively letting users share their context-aware contents everywhere, every time and with any device. As examples, let us consider applications like FourSquare (www.foursquare.com), to share one’s own geographical position on Facebook or Twitter; Shazam (www.shazam.com) to recognize songs by humming on smartphones microphones, or Layar (www.layar.com) to augment reality which has been captured with mobiles cameras.

Participatory sensing (PS) is meant as the possibility to exploit such an endless, interconnected, context-aware community of users in order to gather and share sensed data from surrounding environment. Let us notice that specific sensors, beyond those ones we enumerate above, can be connected to smartphones (typically via Bluetooth or USB). On the one hand, civic benefits are generated from such an environmental monitoring paradigm. On the other hand, instances and challenges about credibility, privacy, consistency are to be faced, together with technological issues underlying such an approach to data gathering (see [3] for the big picture). In general, PS involves a technological and theoretical variety of heterogeneous aspects. Topics on network architectures, from infrastructure-based ones, to mesh ones, up to mobile sensor ad hoc networks must be taken into account to provide a suitable context in sharing data. Processing and network-related restrictions (e.g. to avoid invasiveness) as well as power limitations, which are typical of acting sensors, represent a further aspect to be analyzed. Since some final, unique collector must be conceived - at least to synthesize and fuse data coming from multiple sources – suitable server-side applications, (typically addressed to Web 2.0 mashups and service-oriented applications) are to be designed and implemented as well. Techniques and models of data fusion are finally to be exploited, in order to guarantee consistency and provide credibility. Starting
from the NoiseHound System, which has been designed and implemented to collect noise pollution data from a community of Android smartphones, this paper proposes a protocol called s2s (sensor to sensor), to share measurements among sensing smartphones, depending on their vicinity. The well known Kademlia protocol has inspired the features of s2s as we will detail in the next sections. Improvements in measures accuracy and a smaller overall throughput are achieved by means of a peer-to-peer approach, which is geographical localized to close sensors. Besides assessments with a community of real devices, simulations of s2s have been done to verify our assumptions and point out pros and cons of s2s, the remainder of this paper is organized as follows: section 2 reports background and related works; section 3 depicts the NoiseHound system as the context for s2s; section 4 details s2s; section 5 reports results about real-world assessments and simulations. Finally, section 6 summarizes conclusions and future work.

2. Background And Related Works

Exposure to noise pollution have a well known impact on public health, urban deterioration and social relationships. [4]. From inducted hearing impairment to disorders related to stress, a wide variety of pathologies have been related to noise exposure. Costs in terms of health-care and loss of individuals’ productivity – which is an inducted effect of damages from noise exposure - have been investigated as well, thereby implying National and transnational regulations to be enacted. Prescriptive definitions about sound measurement and practices to assess it have been provided, together with a suitable set of parameters (frequencies, persistence, location, night/day measurement) [5][6]. Effects of noise on objects, such as monuments, streets and buildings are well known and a plethora of solutions have been proposed as architectural techniques and devices [7]. Summarizing, environmental noise is a pathogenic agent and a valuable parameter whose architects, town planners and restorers have to take into account. Dedicated devices (sometimes called phonometers) are generally used to measure sound pressure level (SPL) from their surrounding environment, despite some applications for smartphones recently appeared on the market. The latter ones leverage the microphone to acquire noise samples and compute SPL. The critical aspect of noise sensing is related to the calibration and precision of microphones on meters. Other parameters which describe and make complex on-field noise assessment are related to the context, such as the wind direction and the presence of sound barriers (buildings, bridges, etc.). In particular, models about noise diffusion in urban areas have been proposed (as, e.g. in [8]) to qualify measures on a realistic scenario. Geo-localization and networking capabilities, which are so common (other than, obviously, a microphone) into nowadays smartphones have encouraged the development and assessment of distributed, participatory systems to sense noise pollution at relatively low costs. NoiseTube [9] NoiseSpy [10] and EarPhone [11] are projects utilizing and investigating such an approach. Indeed, the participatory aspect, considering the presence of a certain community of sensing units rather than single devices on-field, is not taken into account as a computational, resource-saving and quality improving feature in these works. A better (or also, consistent) precision in measurements and in calibrating devices can be achieved by suitably fusing and processing data from overlapping measures.

The presence of trustable devices (as terms of accuracy), based on their hardware characteristics, can be used to adjust measures. As an example, let us suppose to know the precision of a certain smartphone model (e.g. its measure average error), then we can provide a feedback adjusting any other device of the same model. Furthermore, trustable measures can be applied also to devices of different model. Finally, frequency of measures can be turn into any dynamic scheduling, in terms of sensing and sharing (i.e. transmitting), according to the number of close devices and their known precision. Data fusion (DF) is a critical step in designing a wireless sensor network as it handles data acquired by multiple sources [12]. Indeed, DF has been investigated to be essentially applied in noisy or power-expensive communication from nodes to nodes/sinks, but challenges and results can be applied to sensing issues as well. In other words, techniques to achieve power saving and reduce error (or improving QoS) in data sharing among hops, can be applied to refine reliability of sensed data, once they are fused on some centralized system. Estimation and inference of acquired data can be considered the main goals for improving participatory sensing. According to the classification presented on [12], cooperative and complementary paradigms for DF are needed, due to the necessity of providing, respectively, a global mapping of noise diffusion over an area and a realistic (local) measure of it by exploiting multiple sensing. Let us consider local and global views of participatory sensing: on the one hand, transmission and receiving are usually more power-consuming than processing on mobiles [13], hence strategies to locally filter data (thereby limiting useless transmissions) are desirable. On the other hand, centralized processing services can provide a faster analysis of data and their knowledge about the sensors network - as a whole - can guarantee a more effective feedback to the nodes. The assumptions above have led us to design and investigate a system which were capable to reduce the amount of information needed to improve the quality of noise sensing on a community common smartphones moving around an urban area. An hybrid network architecture (p2p and client-server), composed of a central server unit in charge of coordinate, set up, fuse data, and some peer-to-peer clusters of geographically close sensors to locally share their measurements. While client-server paradigm let the participatory community to initially organize itself and provide suitable reports of gathered data (through Web 2.0 mashups), the peer to peer approach support optimization in terms of transmitted data (to the server), sensing opportunities (useless/useful) and quality of measurements (see the assumption at the beginning of this section). In particular, The Kademlia protocol [14] have inspired s2s, our p2p protocol to locally find and share information about sensed data by physically closed sensing units. In the next section we will detail the NoiseHound system which s2s has been designed for.

3. NoiseHound issues before s2s

We initially designed and implemented NoiseHound (NH) as a three modules system over a typical client-server network architecture: the NoiseHound application (NM), the NoiseAccert service (NA) and the NoiseReport Web resource (NR). NM is an application to be launched on smartphones equipped with Android OS. It let geo-referenced noise sensing, by acquiring sound pressure level (SPL) and sound frequencies values (SFV) from the surrounding environment through the phone microphone. NA is a server side software that receives SPL and
SFV data from smartphones, together with geographical and temporal coordinates, then providing some processing on them. NR is in charge to mash up processed (fused) data with mapping services (Google maps). In particular, NA is a server-side service which provides further information about noise or makes more effective the contribution of mobiles. NR reports data on the Google streets maps as overlaying, tooltipped, “bubbles”, whose opacity depends on noise level. On the first release of NH, NA was used to: (a) acquire coordinates of each node and feedback it with the opportunity of sampling and sensing data, according to the model detailed on next subsection; (b) calibrate sensors, once a trustable value had been assessed for a certain area at a certain time. NA also provided initial synchronization of nodes sampling (see subsection 3.1), in order to reduce the number of needed sensors. NM schedules a series of sampling sessions in time. In particular, a background process is launched so as to periodically sampling noise. Synchronization of NM with the NA timestamp was initially designed and implemented to improve spatio-temporal coverage of smartphones over the same area (see subsection 3.2).

The first time users launch NM, they are warned to do it on a silent place, to calibrate a zero-noise value. Our system (with its initial, pure client-server architecture) has been assessed with a small community of users equipped with an heterogeneous set of Android smartphones. Different situations have been randomly encountered or forced and they substantially validate our hypothesis, assumptions, and proposed solutions. The next step has been the design and implementation of s2s, a p2p protocol to free NA from providing feedback about a logically local situation. Our aim has been addressed to provide a better balancing among sensing and processing components, thereby moving from a client-server architecture to an hybrid one. The well known Kademlia protocol inspired s2s, as we will report on the next section. Since we can hypothesize a low churn rate as referred to small areas and (typically) pedestrian users, we assumed to overcome limitations of systems like Kademlia on a mobile scenario, by considering variance of items moving into/out to be acceptable for such an approach.

3.1 Estimation model

Any geographical point around a given measurement \( \overline{X} \) (meant as a vector) was assumed to show an SPL that is a random variable \((r.v.)\) \( \bar{X} \) with a density depending upon its distance from \( \overline{X} \). Every other sensor, close enough to any measure \( m_i(\Delta t, \overline{X}) \) and making another estimation, say \( m_j(\Delta t, \overline{X}) \) during the same time interval, can condition the probability of \( \bar{X} \) on every intersection-point (if any) on the areas around \( m_i \) and \( m_j \). In particular, for each geographical point \( \overline{P} \) over an area (except those one where data have been sensed), we assumed its SPL to be an r.v. with a normal density \( MN = N(\mu', \sigma'^2) \) where \( \mu' \) and \( \sigma'^2 \) are calculated on sensed values around \( \overline{P} \) as:

\[
\mu' = \frac{1}{n-1} \sum_{i=1}^{n} m_i(\Delta t, \overline{X})
\]

\[
\sigma'^2 = \frac{1}{n-1} \sum_{i=1}^{n} (m_i(\Delta t, \overline{X}) - \overline{P})^2 - \mu'^2
\]

The factor \( \frac{1}{n-1} \sum_{i=1}^{n} (m_i(\Delta t, \overline{X}) - \overline{P})^2 \) is intuitively related to the average closeness of considered n-1 sensors.

Equations above let the sensing nodes to:

- avoid useless measurement and transmission of sensed data, according to an over-threshold estimated measurement;

- (re)calibrate their zero-noise values on the strength of some inconsistencies with average values around them; acquiring a sort of awareness about any “softening state” due to the typical sound insulation on bags or pockets. In particular, at the beginning of each sensing session, NM communicates its geographical position to NA, which feedbacks a yes/no response about the opportunity of sensing and sending data, based on its calculated \( \mu' \) and \( \sigma'^2 \). After SPL and SVF values are sent, NA can send back information about consistency of measures, by simply comparing \( \mu' \) as calculated without the acquired measure and the measure itself.

3.2 Timing measures

In order to be less invasive as possible, NA periodically (namely, not continuously) tries to sense geo-localized noise. In particular, every 2 minutes NH searches for a GPS (or Network) service for 40 seconds time to get its coordinates, and, if found, it, in case, after receiving the authorization by NA, samples and computes noise for 20 seconds (in the following, we will call this procedure as search-and-sample period). In order to minimize the number of necessary mobiles sensors over a certain zone, we supposed a sort of ordered, vicinity-based synchronization was done for every device at server side. In particular, we considered the number of necessary devices in covering a certain zone as depending from the probability to be synchronized and to remain on the same area for relatively long periods (in comparison with time intervals among search-and-sample sessions. According to such assumptions (which is beyond the scope of this work), NM sets up its first search-and-sample period between a couple of fixed instants according to its vicinity at another device which have been already set up. Depending on a time-arrival ordered list on NA, the first device will start its search-and-sample period in 20 seconds, the second device which is found close to the first after 40 seconds, the third one after 20 and so on, alternatively. This way, every device likely launches its sampling period contemporarily or just after the end of any other one. This approach let the number of sensor to be reduced and is computationally very simple.

4. The s2s protocol – design issues

Instead of exchanging information with NA – either about the opportunity of sensing and sending SPL and SVF, or about inconsistencies of their measures - NMs should logically share such information locally. They essentially could obtain the same
information which came from NA by somehow querying their neighbors. Such an evidence led us to design, implement and assess s2s, which provides mechanisms and rules for the goals above. In the following, we will describe s2s design issues in analogy with those ones from Kademlia, as we assumed it as a starting point for our work.

### 4.1 From DHT to NPT

Kademlia nodes IDs are turned into geographical coordinates on s2s. Each NM communicates such coordinates (sai: gID) to NA at its launch together with its IP address and UDP port. Since, indeed, geographical position can relatively rapidly change in time, gID is to be considered as a sort of tag for an accepted geographical area. Once gID is sent, NA replies NM with a Neighbors Position Table (NPT), which is a list of quintuples <IP address, UDP port, gID, SPL, time_of_measurement>, of all the nodes (recently) being under a certain distance from it. NPT is organized on NM as an ordered circular array of lists as follows: each element from NPT is placed on a list which contains k (k is a constant) triples with a geographical orientation between α and α + s as regard to NM, where \( S = \pi / r \) and \( r \) is a constant. Each list is indexed form 1 to \( r \) so that the lists can be seen as to be placed clockwise from north to east and back to north. Finally, each list is ordered so that least-recently seen neighbors are placed on its head, while most-recently seen nodes are placed at tail. In analogy with Kademlia, each node of NH holds its k-buckets which are ordered according to their relative orientation as regards to the node. As for the analogy with Kademlia, Distributed Hash Table (DHT) is turned into NPT in s2s.

#### 4.2 NM lifecycle with s2s

After its initialization, the typical lifecycle of each NM node in NH can be synthesized as follows: each time GPS coordinates are acquired (at the beginning of a search-and-sample period), all its neighbors are queried about their measurements. If the already done estimations satisfy a threshold, according to the statistical model hinted in the previous section, such a measurement is stored and sent to NA (labeled as approximated). Otherwise, SPL and SVF are gathered, locally stored and sent to NA, together with geographical and temporal coordinates. The lookup measure phase provides all the updates on NPT as follows:

- Replying nodes send their last measurement (if it has been really done and enough recently) and all the data from their NPT which are on slides towards the querying node, that is from the \( r / 2 \leq q \) item on their NPT (See figure 2.).

The query phase ends up when one of the following condition is verified: i) a threshold about the number of useful (in time) measurements is surmounted; ii) all the neighbors from NPT have been queried. Condition (i) leads to the storing of the surrounding measurements as it had been done by the node; condition (ii) implies the sensing, storing and sending of SPL and SVF together with geo-temporal coordinates. S2s consists of three basic operations (and not 4 like Kademlia): PING, EVAL and FIND_NODE, which correspond to PING, FIND_NODE and FIND_VALUE of Kademlia. PING probes a node to see if it is online and (differently from Kademlia) if it is close yet. FIND_NODE takes its NPT item key (say: q) as an argument (other than the IP address of a target node). The recipient returns its neighbors (as \(<\text{IP}\_\text{address, UDP\_port, gID}>\) triples) from the \( r-q \) item, (i.e. from the slide of its surrounding nodes toward the querying node).

```plaintext
for i = 1 to r // scanning circular NPT
    if (PING[scan_NPT(i)] != reply) // doing nothing.
        reply = NULL
        new_neighbors = FIND_NODE[1, reply.IP]
        add_neighbors[new_neighbors];
        if is_consistent(reply.measure)
            amount_of_measurement++;
            Add_to_my_estimation(reply.measure);
        if amount_of_measurement > THRESHOLD
            i = r // exit from query phase;
            success = true;
            send_measure[AS_APPROX,...];
        }
        if (!success)
            sense_store_send();
```

Figure 2. Querying schema between 2 NMs

placed clockwise from north to east and back to north. Finally, each list is ordered so that least-recently seen neighbors are placed on its head, while most-recently seen nodes are placed at tail. In analogy with Kademlia, each node of NH holds its k-buckets which are ordered according to their relative orientation as regards to the node. As for the analogy with Kademlia, Distributed Hash Table (DHT) is turned into NPT in s2s.

**Figure 3. FIND_VALUE procedure for s2s**

FIND_VALUE implements the query phase described above. Algorithm of any query is sketched below on figure 3. Let us notice that approximated measures (i.e. coming from neighbors and not locally sensed) are labeled (with flag AS_APPROX on send_measure()), and locally discarded. This way, in case of queries by other nodes, they are not considered. K-buckets (or slides on the circular array) on NMs are updated with the same policy used on Kademlia (least-recently seen), except for the criteria which is used to push elements from lists. They are periodically eliminated depending on relative displacement of the node.

### 5. S2s Simulation on PeerSim

With the aim to compare the total amount of messages NA receives with or without s2s, we provided an extension of PeerSim [15] with s2s, adding some procedures to generate periodical events for sensing simulation. Each node is
dynamically affected by some search-and-sample periods in which it receives a simulated couple of geographical coordinates. Then it launches its s2s query phase, updates its NPT and sends (in case) its measurement (also simulated) to a particular object we created to emulate the NA presence. Another observer (as a sort of monitor is called in PeerSim), outputs the total amount of approximations and the total amount of actual measurements. Table 1 shows the average percentage of approximated measures in comparison with all the samples to be taken, with 125, 250, 500 and 1000 nodes, which are simulated to randomly move (at a pedestrian/biker typical speed) on an area of about 2 Km². We assumed a threshold of about 125 meters to compute approximation, so that the number of nodes represent the probability of find a suitable number of neighbors around a sensing node. Figure 4 shows the graphics of quasi-linear growth of such percentages.

Let us notice that all such measurements are now in charge of any NM, rather than on NA. Indeed, besides a better load balancing among components (both mobiles and central server), s2s lends itself to be re-arranged inside an ad hoc protocol, as mobile sensor networks are ideally conceived. The percentage referring to 1000 nodes and 3 neighbors (41.49%) should be seen as a realistic condition, since it simply implies a high probability to find 4 sensing nodes on a range of 250 meters. Such a context is quite acceptable to a credible participatory assessment on real-world scenarios.

Table 1 – average percentage of approximations with different thresholds of needed neighbors and different amounts of nodes

<table>
<thead>
<tr>
<th>Nodes</th>
<th>3 nobs</th>
<th>4 nobs</th>
<th>5 nobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>2.25%</td>
<td>0.10%</td>
<td>0.05%</td>
</tr>
<tr>
<td>250</td>
<td>9.25%</td>
<td>4.35%</td>
<td>1.44%</td>
</tr>
<tr>
<td>500</td>
<td>26.48%</td>
<td>14.85%</td>
<td>7.31%</td>
</tr>
<tr>
<td>1000</td>
<td>41.49%</td>
<td>28.15%</td>
<td>18.24%</td>
</tr>
</tbody>
</table>

Figure 4 – flows of approximation percentages (see Table 1)

6. Conclusions

S2s has been conceived to improve messages exchanging and information processing into NoiseHound, a participatory sensing system we previously built to monitor noise pollution by means of common smartphones. Kademia DHT and primitives have inspired s2s in terms of update and lookup procedures to localize decisions about the opportunity of sensing/sending measurements. PeerSim has been chosen as a simulation environment to a large-scale evaluation of our hypothesis, since it provides all the features to obtain a down-to-earth assessment. We have also implemented a prototype of s2s on Android platforms, expecting to obtain meaningful case-studies with real-world suitably great communities of users. Limitations in the number of users we provided with NoisHound+s2s and relatively high costs of connection to the Internet are limiting assessments yet. A Further step we are investigating is addressed to exploit Bluetooth networking and ad hoc (through wi-fi interfaces) connections to surmount current limitation in connecting mobile sensing nodes.

7. REFERENCES


