Demo Abstract: Exploring Smartphone-based Participatory Computing to Improve Pervasive Surveillance

Zheng Dong[1], Banghui Lu[1], Liang He[1], Peng Cheng[1], Yu Gu[1] and Lu Fang[2]
[1]Singapore University of Technology and Design, Singapore
[2]University of Science and Technology of China, China

ABSTRACT

Participatory Computing is a promising solution to fully utilize the wasted computation resources of mobile devices such as smartphones. In this demo abstract, we present our design and implementation of an participatory computing enhanced pervasive surveillance system. Our evaluation results show that the proposed system can effectively utilize the computation capability of mobile devices while guaranteeing the service reliability even with the intermittent nature of participatory computing.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design

General Terms

Design, Experimentation, Performance

Keywords

Participatory Computing, Task assignment

1. INTRODUCTION

In this paper, we propose the modeling, analysis, and implementation of *participatory computing*, where the computational power of mobile devices can be leveraged to execute sensing tasks that are computational expensive. To realize efficient participatory computing, we have to tackle several critical research challenges. First, different from traditional computing devices, computational power on mobile devices such as smartphones have highly unpredictable and intermittent availabilities. Moreover, many sensing tasks such as market price tracking and real-time crime search are delay-sensitive and often possess a tight response time requirement in the range of a few minutes or seconds. Last but not the least, different from many distributed systems that achieve fault tolerance by checkpointing or replication, which either incurs high task management cost or high overhead for individual machines, system reliability in participatory computing has to be ensured in a way such that overheads at individual participating devices are minimized.

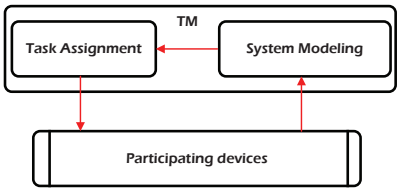


Figure 1: REPC Architecture

Motivated by above observations, in this paper we develop a simple yet highly reliable and efficient participatory computing framework, which we call REPC, to achieve reliable and timely computations for the participatory computing with close to optimal overhead based upon the real-time statistics of participating devices in the system. Our results demonstrate that even though participating devices may join or leave the system intermittently with limited durations, our design is able to guarantee a specified threshold of completed tasks in the task pool while minimizing the number of tasks executed on individual participating devices.

In the demonstration, we implement a participatory computing system with the application scenario of pervasive surveillance to demonstrate the effectiveness of the proposed task assignment framework. In this application, the camera needs to find out whether any wanted criminal appears in its footage. Without sufficient computational capabilities and central servers, the camera recruits nearby participatory computing devices such as smartphones to assist the image processing tasks and thus facilitates the criminal detection.

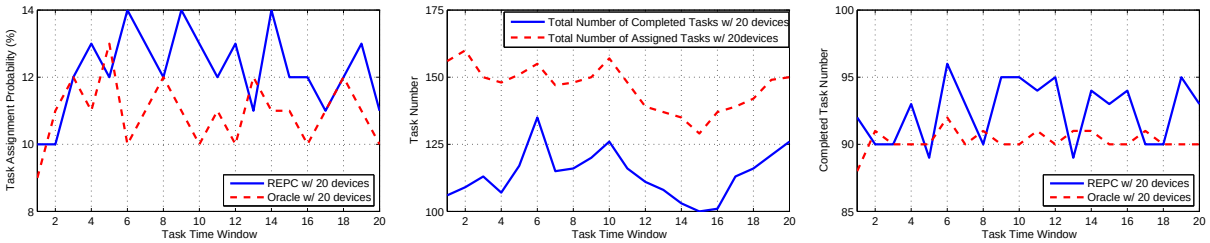
2. FRAMEWORK OVERVIEW

The basic architecture of REPC is shown in Fig.1. Specifically, the Framework is composed of Task Manager (TM) and participatory devices. Typically, the TM estimates the statistics of participatory devices in a real-time manner (i.e., System Modeling). Based on the online estimation, the TM dynamically assigns image processing tasks to each individual participatory device (i.e., Task Assignment). The tasks assigned to individual devices are desired to be 1) as many as possible to fully utilize the devices' computation capability and reduce the detection delay, and 2) not too much to over-occupy the devices' capability, in which case they cannot reply any valid processing results.

2.1 Basic Task Manager

In our framework, the participating devices can locally decide whether or not to accept the footage sent by TM

SenSys '13, Nov 11-15, 2013, Rome, Italy
ACM 978-1-4503-2027-6/13/11.



(a) Task Assignment Probability (b) # of Assigned Tasks (c) # of Unique Completed Tasks

Figure 2: System Performance With 20 Participating Devices

and can abort the execution of assigned tasks at any time. Therefore, to ensure all tasks at TM can be successfully computed by the participating devices and returned to TM with high confidence, the TM must decide which task will be executed on which participating device and make proper redundancy.

Suppose that at any specific time, TM has a task set $\mathcal{T}$ with n tasks. TM randomly assigns a task to a participating device with a probability p, and any M devices return the assigned tasks successfully, then the expected number of completed tasks in the task set $\mathcal{T}$ can be calculated by

$$E(\mathcal{T}) = \sum_{i=1}^{n}(1-(1-p)^M) = n(1-(1-p)^M). \quad (1)$$

2.2 Participatory Devices

To guarantee the expected number of completed tasks in equation.1, we design a feedback mechanism to estimate the statistics of participating devices. In our framework, we use λ and μ to represent the arrival density and departure density of participatory devices. By recording the number of participating devices entering the system within a predefined time window t_w, the task manager (TM) is able to estimate the arrival density λ. Similarly, through periodic beaconing [1], TM is also able to estimate the departure density μ.

Based on the queueing model, the relationship between $E(\mathcal{T})$ and task assignment probability p can be denoted as

$$E(\mathcal{T}) = n \times (1-(1-p)^{\frac{\lambda t_w}{e^{\mu p n t_c}}}), \quad (2)$$

where t_c is the average computation time of a footage matching. To minimize the workload on each participatory device, our system need to find the minimum p for a required $E(\mathcal{T})$. According to equation.2, we can prove that the minimum p belongs to an interval where $E(\mathcal{T})$ is monotonically increasing, and thus we use binary search to get the minimum p.

2.3 Energy Overhead Analysis

Energy is one of the most precious resources for smartphones. In the following, we use a quantitative example to show that although additional computation and communication loads are introduced by REPC, current smartphones are well capable to handle the associated energy consumptions.

Let us take the Sumsung Nexus S with a $1,500\ mAh$ battery as an example. If the participating devices are only willing to contribute 1% of their capacities, an amount of $1,500 \times 1\% = 15\ mAh$ capacity is available for REPC. The current draw for the WiFi module's receiving state of Nexus S is about 117 mA and that for its CPU processing with the peak frequency is 150 mA. As a result, the provided 15 mAh capacity can support REPC to execute at its full frequency for least $\frac{15\times3600}{117+150} \approx 202\ s$. This time is far sufficient for REPC's time window ($\approx 15\ s$ according to our experiments), and thus alleviates our concern on the energy overhead.

3. SYSTEM IMPLEMENTATION

We implement a REPC-based surveillance system prototype which will be demonstrated. The TM module is developed using Qt SDK on a Linux laptop. We emulate the scenario that a surveillance camera periodically captures images that may contain the face pictures of wanted criminals. Therefore, a number of image matching tasks are generated and distributed to participating smartphones according to the task assignment scheme in our design. Each participant holding a smartphone moves in the open area randomly and freely without restrictions. Participating devices are connected to the laptop through a wireless access point.

We also implement the task execution module on Android phones, including an image matching function using OpenCV4Android SDK. Once a smartphone receives a task request, it performs image matching to check whether the query image matches any image in the local database, and returns the result to the surveillance camera.

TM adopts the time window based operation schema to divide and assign tasks. Whenever the entrance of a participating device is detected, the TM randomly selects a ratio p of total tasks from the current window and assigns them to this device. If the assigned tasks are accomplished before the corresponding device leaves the system, they will be returned to the TM. Otherwise, these tasks will be discarded by the device and treated as uncompleted.

Fig. 2 presents the system performance over time. Comparison is conducted with the Oracle task assignment. The Oracle task assignment is defined as that tasks are assigned based on the traces of these participatory devices, which is not available in reality. Oracle task assignment is the lower bound of overhead. Results demonstrate that our system approaches about 90% to the Oracle task assignment.

4. ACKNOWLEDGMENTS

This research was supported by Singapore-MIT International Design Center IDG31000101 and iTrust Cyber Physical System Protection project.

5. REFERENCES

[1] R. Zhou, Y. Xiong, G. Xing, L. Sun, and J. Ma. ZiFi: wireless LAN discovery via ZigBee interference signatures. In *Mobicom'10*, 2010.