2015 16th IEEE International Conference on Mobile Data Management

SmartSource: A Mobile Q&A Middleware Powered by Crowdsourcing

Ye Zhao[1], Chen-Chih Liao[2], Ting-Yi Lin[2], Jikai Yin[3], Ngoc Do[4], Cheng-Hsin Hsu[2], and Nalini Venkatasubramanian[3]
[1]Google Inc., Mountain View, CA, USA
[2]Department of Computer Science, National Tsing Hua University, Hsin Chu, Taiwan
[3]Department of Computer Science, University of California Irvine, CA, USA
[4]Arista Networks, Santa Clara, CA, USA

***Abstract*—In this paper, we introduce SmartSource, a crowdsourcing based mobile Question & Answer (Q&A) system that aims to provide mobile information seekers with timely, trustworthy and accurate answers while ensuring that information providers are not inappropriately burdened. We tackle this challenge by taking advantage of both static and dynamic context and semantics from mobile users (e.g., geolocation, social network, expertise/interest, device sensor profiles, battery level) to identify sources of information (i.e., workers) that are trusted by the user and accurate enough for the questions at hand. Given a question, the SmartSource broker middleware executes a scalable and efficient worker selection algorithm that uses a Lyapunov optimization framework to maximize the utility of worker selection while guaranteeing the stability of the overall system. An associated assignor selection is used to scale the selection process to a large number of users. We implement the SmartSource prototype system on an Android testbed and thoroughly evaluate the system using real world applications and data, in particular those that involve geospatial questions and answers. Evaluation results indicate that SmartSource is efficient and provides superior worker selection compared to baseline approaches. SmartSource is also highly customizable: it employs a general utility function and provides a control knob to tradeoff the optimality and responding time. We believe that SmartSource will pave a way for new mechanisms of interaction among mobile users.**

***Index Terms*—Q&A systems, middleware, mobile computing, crowdsourcing, crowdsensing, performance optimization**

I. Introduction

Mobile devices, such as smartphones, tablets, glasses, and smartwatches, are getting extremely popular in our daily life. For example, a market report [4] points out that more than 70% of Americans have access to smartphones as of 2014 - this allows for anytime, anywhere access to information through the mobile Internet for many users. We conjecture that this rapid increase in smartphone penetration worldwide will enable a new level of interactivity in information exchange. In particular, we anticipate that the time is ripe for the rise of an *interactive Mobile Question and Answer (Q&A) ecosystem* that dynamically connects information seekers (i.e., humans with questions) to the best possible information providers (other humans who can provide timely and accurate answers). Unlike existing information retrieval systems (e.g., search engines) and web-based *Question and Answer (Q&A)* systems (e.g., AskJeeves), we argue for a novel mobile Q&A paradigm that provides fine-grained information that is personalized to the current needs of the information seeker. The envisioned mobile (Q&A) system will allow users to ask specific questions, such as: 1) "Is it raining outside Bren Hall right now?", 2) "What's the current south-bound speed at exit 7 of I-405?", and 3) "How long is the line at the on-campus MacDonald's?"; and to use a combination of methods (including responses from multiple other users) to provide answers to these questions. In this paper, we propose such a flexible and interactive mobile Q&A middleware, called SmartSource that intelligently leverages crowdsourcing mechanisms to provide accurate and timely responses to questions.

Adapting search-driven Q&A systems in mobile settings is fairly challenging, primarily due to highly dynamic and diverse mobile user needs and contexts. For example, propagating a question about the road conditions near Corona Del Mar Beach to a broader population, e.g., mobile users in Orange County, California is not useful. Careful selection of the target recipients of a question is essential to (a) ensure that users can participate meaningfully (indeed, noisy apps are typically turned off or deleted by the user) and (b) lower resource consumption in the end-devices and networks. Additionally, mobile users may only want to receive answers from other trusted users such as his/her Facebook friends who are capable of responding.

The use of *crowdsourcing* as a driving mechanism to build Q&A systems is promising. Crowdsourcing refers to public platforms (e.g., Amazon Mechanical Turk (AMT) [1] and CrowdFlower [2]) that allow users to hire others to perform certain tasks; example applications include voting systems, image retrieval and assesment of image searches [37], multimedia annotations, information sharing, and social games. Enabling such crowdsourcing in mobile devices [8], [21], [37], [32], [17], [38], [9], [10] has been a growing area of study. A recent large-scale measurement study with 85 smartphone users [8] reveals that a small number of smartphone users can provide an impressive coverage of a big city. Recent work in *crowdsensing* explores the use of on-board and external sensing mechanisms, including efficient user localization [32] for a range of applications such as air quality monitoring [17], determining transportation times [38], building occupancy [9] and parking spot availability [10]; such content may further be piggybacked on other messages, calls or applications for efficient resource usage [21]. Several of the crowdsourcing examples above are intended to be used in a highly asynchronous

978-1-4799-9972-9/15 $31.00 © 2015 IEEE
DOI 10.1109/MDM.2015.73

manner where responses may arrive in the order of hours/days; the crowdsensing apps on the other hand are intended to work without any human intervention.

In contrast to the above efforts, the crowdsourcing based mobile Q&A system must be general purpose (usable for multiple applications), interactive (exploit human-in-the-loop), dynamic (near real-time responses in an asynchronous setting where participants join and leave at will) and opportunistic (aims to use the best possible responders for the question available). The *SmartSource mobile crowdsourcing middleware* achieves the above requirements through a combination of strategies. First, SmartSource provides relevant and trustworthy answers through careful selection of responders (e.g., nearby users for location based questions). Secondly, SmartSource enables dynamic/flexible interactions by allowing mobile users to ask/answer questions anywhere and anytime, and choose whose questions to answer and whose answers to take. Finally, the SmartSource implementation supports cost-effective deployment by reducing end-to-end system overhead (e.g., do not distribute questions to less relevant responders) and being cognizant of device capabilities and resources (e.g., residual battery capacity). More detailed comparisons among SmartSource and the work in the literature can be found in Section VI.

Key contributions of this paper include:

- Design of a generic mobile Q&A architecture that brings together information seekers and providers to answer a broad range of questions that involve spatial-temporal context and expertise of users.
- Formulation of optimization problems (cast as worker selection and assignor selection problems) to balance the multiple design goals of accuracy, trustworthiness and overhead cost, at scale.
- Design of stable algorithms for worker selection and assignor selection using the Lyapunov queuing framework [13], [15].
- Implementation of a prototype SmartSource system on a smartphone testbed and evaluating it through both user studies (for usage) and simulations (for scalability).

II. Middleware Design Overview

In SmartSource, a key design problem is how to target potential information providers to answer questions so that providers are not burdened inappropriately and information requester can obtain trustworthy and timely information. The solution lies in identifying sources of information that are trusted enough for the question at hand; the choice of information sources uses multiple factors including : a) Experts - individual with expertise in the question domain are more trusted when questions are knowledge-driven; b) Friends and family networks - A requester is more likely to trust the answers coming from a close friend or family member as compared to a random responder; c) Those in the vicinity of a location are potentially more suitable candidates to answer location-based questions; and d) A combination of the above. Furthermore, because some questions may require to be answered by sensors embedded in smartphones of users who act as information providers (e.g., pollution sensors can be used to answer questions such as "how's the air quality?"); in such cases, we need to target mobile users equipped with capable sensors and enough residual energy to answer the questions.

SmartSource uses both static and dynamic contexts and semantics from mobile users (and their smartphones) to recommend potential information providers for questions through a novel utility-driven worker selection algorithm. The contexts include: mobile users' geolocation, social network connections, expertise and interests, profiling of device sensors, and device battery levels. Moreover, our design keeps scalability and stability in mind. We accommodate a distributed environment, and develop fine-grained control mechanisms based on the Lyapunov optimization framework [13], [15] that maximizes the utility of worker selection while maintaining a stabilized load of the middleware. We present an overview on the design in the rest of this section.

A. SmartSource Architecture

Figure 1 depicts the overview of the SmartSource architecture. It consists of querists, workers, and Q/A broker(s).

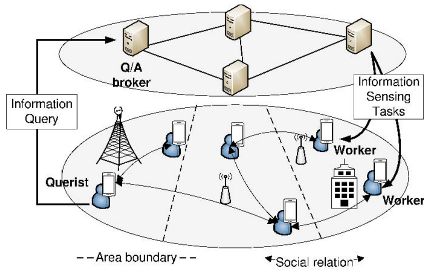


Fig. 1. Overview of the SmartSource system.

Querists: Querists are registered users. They send queries to the broker system to request for specific information they want know. This can be done through either a mobile app or web portal of the system. In general a query may consist of: 1) question, 2) location and time specifications, 3) requirements, and 4) reward. The question indicates what information the querist wants to get. Some questions may be location specific, such as "How many sea lions are at Pier 39?". The location specification may be a stadium, a park, a room, or (more general) a tuple of longitude, latitude, and altitude. Also questions have time specification since querists usually request for information at current time or in near future. Time specification indicates the time frame of the requested information, such as "within an hour from now" or "from 1pm to 5pm". Requirements can be the quality and quantity of the requested information such as the accuracy and the number of answers that the querist desires. In crowdsourcing applications, querists may put reward (as promised monetary or virtual benefit) into their queries to motivate unknown workers to work on their queries.

Workers: Workers are registered users as well. Typically they are mobile users with a mobile app of the system installed on their smartphones. In Q&A and crowdsourcing applications, workers can receive and decide what queries to answer or to work on in three ways: 1) Q&A applications publicly publish submitted queries onto web portals and workers can browse and search questions to answer; 2) to improve targeting, many Q&A applications incorporate a subscription service where workers subscribe to interested topics/fields and applications present queries matching their subscriptions on the workers' home/wall page; 3) workers notify the applications when they are available and applications send notifications on recommended questions to workers. In this work, we focus on the enabling techniques for the third approach.

Broker(s): The key coordination is performed at a broker - smartphone users need to register with the Q&A applications at the broker for the brokerage service. The broker keeps track of smartphone users' static and dynamic contexts and carefully select potential workers for submitted queries. Depending on a query's requirement the broker may select one or multiple workers to sent the query to. To descriptively differentiate queries submitted to the broker and copies sent to workers in notifications, we call the copies of queries sent to workers as *information tasks*. Since workers may refuse to take the recommended tasks and eligible workers may not exist all the time, the brokerage service needs to re-select workers for unassigned queries when new workers are available.

We illustrate the Q&A brokerage service of SmartSource in an example. Consider that Alice is going to a night market, however, she wants to know which booths are popular and how long are the lines for them. She wants the answers back in an hour, so she sends a query to the broker as a querist. Some registered smartphone users located close to the night market are available to answer questions and they become potential workers of the query. The broker notifies the workers on their smartphones with respect to the task and they interact to accept or reject it. If they accept to answer, they may walk to the night market and take pictures of the booths and lines as answers and return them to Alice by the broker. Otherwise, if they reject the task, the broker keeps looking for new workers before the query expires.

Figure 2 shows the major software modules in the SmartSource broker middleware. The *query receiver* receives new queries from querists and put them into the *worker selection queue*. The *mobile user information collector* monitors the static and dynamic contexts of mobile users and maintains the *mobile user information* and *social relation information* databases. The crux of the middleware is the *worker selection algorithm* that determines the best candidate workers for queries in the worker selection queue based on the query requirements and worker status from the databases. The *task assignment module* interacts with the selected workers to notify them of the queried tasks and let them decide whether to accept the task. The middleware requires workers to confirm the assignment (e.g., by clicking an "accept" button) within a certain period after receiving the notification, if they are willing to provide answers. Otherwise, it is considered as not assigned. The *task assignment module* monitors the acceptance of tasks. Once a query is accepted by enough workers it is removed from the worker selection queue. It is important to maintain a stable size of the worker selection queue to keep the broker middleware operational. In this work, we design the worker selection algorithm such that it maximizes the utility of worker selection while maintaining a stabilized size of the queue.

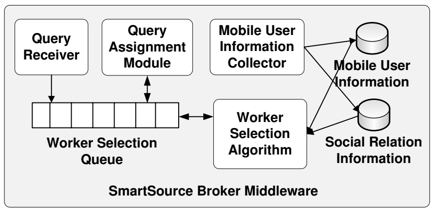

Query Receiver
Query Assignment Module
Mobile User Information Collector
Mobile User Information
Worker Selection Queue
Worker Selection Algorithm
Social Relation Information
SmartSource Broker Middleware

Fig. 2. Software modules in SmartSource broker middleware.

B. Scalability Consideration

Assignors and relays: To support scalable information crowdsourcing to a large number of geographically distributed mobile users, we envision a distributed broker network where each broker takes responsibility for mobile users in a specific region. To be able to flexibly adjust to the distribution of users and provide an efficient structure for information exchanges among brokers, we adopt a distributed hierarchical geo-overlay to connect brokers. More specifically, we apply the RRTree protocol [20] to construct a geo-based nested tree structure consisting of non-leaf brokers, we call them *relays*, and leaf brokers, we call them *assignors*, as illustrated in Figure 3.

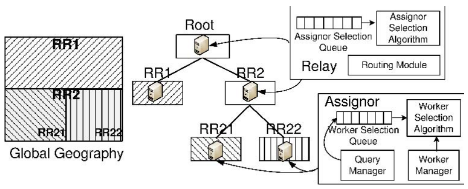


Fig. 3. Geo-based nested tree structure.

In the RRTree structure each node (i.e., broker) is responsible for a rectangular geographical region called Responsible Region (RR), and inherently maintains the following property. A parent RR in the tree subsumes the regions covered by its child RRs, and the root of the tree covers the global geography that is supported by the system. The structure is flexible and scalable, and handles the non-uniform distribution of users gracefully by allowing smaller RRs and denser broker deployment in highly populated locations.

Leaf brokers or assignors, are responsible for receiving queries from and assigning tasks to mobile users located inside their responsible regions. The key function of assignors is worker selection. The software module is the one indicated in

Figure 2. Non-leaf brokers or relays, as ancestors of assignors in the geo-based tree, are responsible for forwarding queries across regions to different assignors for worker selection. We identify following applications that desire forwarding queries to different assignors: 1) in spatial Q&A, if the responsible region of the current assignor does not cover the requested locations of the location-based queries (e.g., querists whose requested locations are not in the same region), the queries need to be forwarded to the right assignors for worker selection; 2) in non-spatial Q&A where queries do not have restrictions on workers' locations, queries can be forwarded from heavily loaded assignors to less loaded ones to achieve load balancing and allow faster query assignment.

The RRTree overlay implements a geographical routing protocol inherent to the geo-aware structure (see [20] for details). We incorporate the routing into the middleware to support query forwarding for spatial Q&A applications. On the other hand, to provide efficient load balancing for non-spatial Q&A applications, we design a distributed *assignor selection algorithm* for relays to determine best assignors for queries. The beauty of the algorithm is that it does not compromise the long term utility achieved by the *worker selection algorithm*.

Figure 3 illustrates the software modules of a relay. The *routing module* incorporates the geographical routing inherent to the RRTree structure to provide efficient query forwarding for spatial Q&A applications. To support scalable non-spatial Q&A with efficient load balancing in distributed environments, each relay keeps track of the backlogs of queues for a randomized subset of assignors in the overlay through the *assignor manager*. This can be easily achieved through peer sampling services such as [29]. Each relay has an *assignor selection queue* to hold non-location-based queries and the *assignor selection algorithm* determines for each query in the queue which assignor it should be sent to for worker selection.

III. System Model and Problem Formulation

We focus on the worker selection problem at assignor brokers first. The assignor selection problem on relay brokers is presented in Section IV-C.

A. System Models

We first present general models used in our system design.

Query model: Let Π be the class of questions. We consider a general information query r from a querist q_r with following optional attributes: 1) $h_r \subseteq \Pi$ as the requested question; 2) g_r as the location(s) of the question; 3) $T_r =< T_r^{start}, T_r^{end} >$ as the time frame of the question in interest; 4) for information collection queries, d_r as the required duration of collection and it must be smaller than the time frame of the question; 5) k_r as the number of workers or answers the querist desires to get; 6) b_r as reward or benefit (money or virtual credit) the querist promises to each worker.

Task model: To send queries to workers, SmartSource first converts queries to information tasks. A task is essentially the same as a query except that each task can be performed by a single worker independently. Worker cooperation and collaboration are out of the scope of this paper. Thus, for a query that desires k_r answers the middleware creates k_r identical tasks and select k_r workers to send them to. Moreover, to ensure that workers only receive tasks that can be answered or worked on shortly, a query is considered inactive for worker selection if the start time of the interested context T_r^{start} is too far in the future. We define a time interval T_θ as the *preparation period* and convert a query to active tasks at the first time instance t when $t + T_\theta \geq T_r^{start}$. Inactive queries are simply cached at their receiving brokers.

Worker model: The middleware maintains up-to-date static and dynamic contexts of workers. A worker w's profile contains the following information: 1) $y_w \subseteq \Pi$ as the capability of the worker to answer various questions. It could be the expertise for knowledge-oriented questions, or the capability of device sensors for information collection questions; 2) $i_w \subseteq \Pi$ as the interest of the worker to answer; 3) l_w as the current location of the worker; 4) E_w as the current energy level of the worker's smartphone; 5) n_w as the number of tasks the worker wants to receive, and $n_w = 0$ means the worker currently can take no more tasks (e.g., because the worker is already occupied); 6) f_w as the desired labor reward per unit time; 7) F_w is the friend circle of the worker, which is derived from the worker's online social networks upon registration. Here we consider 1-hop friends only, but it can be readily extended to include multi-hop friends. Besides the above information, the system also maintains R_w as a list of rejected queries that the worker refuses to work on. This is required by the worker selection algorithm to ensure that the worker won't receive the same task again once he/she rejects it. Let Q_i denote the task queue at an assignor broker i for worker selection. We divide time into slots (e.g., 10 minutes), and model dynamics of the queue at the brokers.

Task arrival: Let $\lambda_i(t)$ be the number of new tasks arriving to broker i's worker selection queue at t. Assume that $\lambda_i(t)$ is finite and limited by λ_{max}, i.e., $\lambda_i(t) \leq \lambda_{max} \forall t$. The time averaged task arrival rate can be written as $\overline{\lambda_i} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\lambda_i(\tau)\}$.

Task-worker selection and assignment: Every time slot, assignor brokers run the worker selection algorithm (presented in Section IV-B) to select workers for tasks in their queues, and send the tasks to the workers. Let $\mu_i(t)$ be the number of tasks at assignor i that have their workers selected at t; let $\mu^r(t)$ be the number of selected tasks of the same query r, $\mu^w(t)$ be the number of selected tasks to the same worker w, and $\mu^{r,w}(t)$ be the number of selected tasks of query r to worker w. We assume $\mu_i(t)$ is limited by μ_{max}, i.e., $\mu_i(t) \leq \mu_{max} \forall t$. Again, the time averaged rate is $\overline{\mu_i} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu_i(\tau)\}$.

A worker, upon receiving a task, decides whether or not to accept the task. If the worker accepts the task, the task is considered as successfully assigned and removed from the worker selection queue. Also, n_w, as the number of tasks the worker wants to receive, is reduced by 1. If the worker rejects the task, the task is kept in the queue and considered for worker selection in next time slot. Also, the query is added to the worker's rejected query list R_w, so the worker will not get the task again. We require the worker to response his/her decision (accept or reject) within the same time slot when he/she receives the task, otherwise the assignment is aborted and the task is considered for worker selection again in next slot. Let $a_i(t)$ be the number of accepted/assigned tasks by assignor i at t. We have $a_i(t) \leq \mu_i(t) \forall t$. The time average rate is $\overline{a_i} \triangleq \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{a_i(\tau)\}$, and $\overline{a_i} \leq \overline{\mu_i}$. We assume $\overline{a_i} \geq \varepsilon \cdot \overline{\mu_i}$ where $0 < \varepsilon < 1$, which can be arbitrarily small. This is to ensure a positive worker selection feedback that at least some portion of the selected tasks can be successfully assigned.

Task dropping: Tasks that couldn't get assigned for a long time in the system or have expired (i.e., the interested time frame of the context has passed) need to be dropped from the worker selection queue. Let $x_i(t)$ be the number of tasks to drop at assignor i at t, and its time average rate is $\overline{x_i} \triangleq \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{x_i(\tau)\}$. Therefore, the evolution of the queue at an assignor broker i is written as:

$$Q_i(t+1) \triangleq [Q_i(t) + \lambda_i(t) - a_i(t) - x_i(t)]^+, \qquad (1)$$

where $[\pi]^+ \triangleq max(0, \pi)$. We define the stability of the queue as:

$$\overline{Q_i} \triangleq \lim_{t\to\infty} sup \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{Q_i(\tau)\} < \infty. \qquad (2)$$

At $t = 0$ we set $Q_i(t) = 0$. The queue is stable as long as $\overline{\lambda_i} \leq \overline{a_i} + \overline{x_i}$.

B. Problem Formulation

Considering a task of query r and a worker w at time t, we first design working criteria to test the feasibility of assigning the worker onto the task. The criteria includes many factors that can be applied to many mobile Q&A applications: 1) $E_w \geq E_\theta$ where E_θ is the energy threshold for a worker to receive new tasks; 2) $n_w > 0$ as the worker wants to receive new tasks; 3) $r \notin R_w$ as the query is not in the worker's reject list; 4) $h_r \subseteq y_w$ as the w's expertise or capability is able to answer the requested question; 5) let $\hat{T}_{r,w}$ be the estimated time duration for w to complete the task of r, and for spatial sensing questions, it can be evaluated as $\hat{T}_{r,w} = d_r + T^{mov}_{r,w}$, where d_r is the required sensing duration and $T^{mov}_{r,w}$ is the estimated time for the worker to move from his/her current location to the nearest requested location, which can be derived from services such as Google Map [3]. Then we require $t + \hat{T}_{r,w} < T^{end}_r$ as the worker has enough time to complete the task before the task expires; 6) $b_r \geq f_w \cdot \hat{T}_{r,w}$ as the reward provided by the querist for the worker who meets the demand.

Utility function: We propose a general utility function of assigning a task to a worker that can accommodate many mobile Q&A applications as follows:

$$U^A_{r,w}(t) = [I_{r,w} + \alpha \cdot S_{r,w} - \beta \cdot M_{r,w} - \gamma \cdot C_{r,w}]^+, \qquad (3)$$

where $I_{r,w}$ is the interest similarity of query r and worker w, $S_{r,w}$ is the social closeness of the querist q_r and w, $M_{r,w}$ and $C_{r,w}$ are the estimated moving cost and energy cost respectively for spatial crowdsourcing applications. Interest similarity $I_{r,w}$ is evaluated using the Jaccard similarity coefficient on h_r as the requested question of r and i_w as the interested question of w, i.e., $I_{r,w} = |h_r \cap i_w|/|h_r \cup i_w|$. The social closeness between the querist and the worker directly affects the willingness of the worker to work on the query and affects the trust of the querist to the worker's work. Recent work has studied how to effectively calculate the social closeness between two users. It can be either explicitly from users' rating or implicitly from users' social communications [26], [25]. We evaluate moving cost $M_{r,w} = T^{mov}_{r,w}$ as the time that the worker has to spend on moving to the desired location(s) of the task. Furthermore, we estimate energy cost $C_{r,w}$ from the energy profile of w's device sensors and the requested sensing question h_r.

To avoid overflowing the queues, in every time slot the system also determines tasks to drop. Dropped tasks will lose the chance for worker selection in the future. We capture the loss of dropping a task of query r at t as: $U^D_r(t) = [d \cdot P_r(t)]^+$, where d is a constant and $P_r(t)$ is time decay function, such as: $P_r(t) = 1 - e^{-(T^{end}_r - t)}$. Thus, the system prefers to drop tasks approaching their deadlines because they have less chance to find workers before expire.

Problem formulation: We consider the following formulation with a long term goal for worker selection in the system.

$$\text{max:} \quad \bar{U} = \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\Big\{U^A(\tau) - U^D(\tau)\Big\}; \qquad (4a)$$

$$\text{st:} \quad \bar{\lambda}_i \leq \bar{a}_i + \bar{x}_i \ \forall i; \qquad (4b)$$

$$\mu^{r,w}(t) \leq 1 \ \forall t; \qquad (4c)$$

$$\mu^w(t) \leq n_w. \qquad (4d)$$

The objective function in Eq. (4a) maximizes the long term utility of selecting workers and minimizes the loss of dropping tasks. The constraint in Eq. (4b) stabilizes the system. The constraint in Eq. (4c) ensures that the system will not send more than one task of the same query to a worker. The constraint in Eq. (4d) makes sure that the number of tasks assigned to a worker is no more than the number of tasks the worker wants to receive.

IV. SmartSource Algorithms

We design optimal algorithms to solve the problem formulated in Eqs. (4a)-(4d) based on Lyapunov optimization [13], [15], which is useful for solving long-term optimization problems. We start by presenting our design principles.

A. Design Principles

We design a novel mechanism to take into account task urgency in worker selection, i.e., to prioritize tasks with approaching expiration deadlines over other tasks. We maintain a dynamic *priority-weight* $u_k(t)$ for each unassigned task k in queue Q_i as an exponential function written as:

$$u_k(t) = c \cdot e^{-(T^{end}_r - t)}. \qquad (5)$$

For a new task k arrives or activates at t_k, its priority-weight is initialized as $u_k(t_k) = c \cdot e^{-(T^{end}_r - t_k)}$. When the task expires at $t = T_r^{end}$, the weight is $u_k(T_r^{end}) = c$. We can write the time evolution of $u_k(t)$ as:

$$u_k(t+1) = u_k(t) \cdot e = u_k(t) + u_k(t)(e-1). \quad (6)$$

We use a novel virtual priority-weight queue Z_i at each broker holding the priority-weights of the unassigned tasks. Once a task is successfully assigned or dropped, its weight is removed from the virtual queue. In practice, the virtual queue is simply a counter as the sum of the weights:

$$Z_i(t) = \sum_{k \in Q_i(t)} u_k(t). \quad (7)$$

Therefore, we can write the queuing dynamics of Z_i based on that of Q_i as follows:

$$Z_i(t+1) \triangleq [Z_i(t) + \rho_i(t) + \lambda'_i(t) - a'_i(t) - x'_i(t)]^+, \quad (8)$$

where $\rho_i(t) = \sum_{k \in Q_i(t)} u_k(t)(e-1)$, $\lambda'_i(t) = \sum_{k \in \lambda_i(t)} u_k(t_k)$, $\pi'_i(t) = \sum_{k \in \pi_i(t)} u_k(t) \cdot e$, and π is a replacer for symbol μ, a and x.

The stability of the virtual queue can be defined as:

$$\overline{Z_i} \triangleq \lim_{t \to \infty} sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{Z_i(\tau)\} < \infty. \quad (9)$$

From Eq. (7), we see $Z_i(t) = 0$ when $t = 0$. Also, the virtual queue is stable when the real queue Q_i is stable and $u_k(t)$ is limited.

Define $\Theta(t) \triangleq (Q(t), Z(t))$ as the concatenated vector of the real and virtual queues. We define a quadratic Lyapunov function:

$$L(\Theta(t)) \triangleq \frac{1}{2} \sum_i (Q_i(t)^2 + Z_i(t)^2). \quad (10)$$

A one-step Lyapunov drift is defined as:

$$\Delta(\Theta(t)) \triangleq \mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t)) | \Theta(t)\}. \quad (11)$$

Intuitively, to stabilize the system, we want to maintain $\Delta(\Theta(t))$ as small as possible, so that $L(\Theta(t))$ is small. This is done by maximizing the number of tasks to assign as well as to drop in a time slot regardless of utility. To take into account the utility and the long term goal we defined in Eq. (4a), we construct the following per-slot objective:

$$\min : \Delta(\Theta(t)) - V\mathbb{E}\{U^A(t) - U^D(t) | \Theta(t)\}, \quad (12)$$

where V is a weight that controls the relevant importance of the utility optimization over the stability of the queues.

From the queuing dynamics of Q and Z (Eqs. (1) and (8)) and the definition of the Lyapunov drift (Eq. (11)) we derive the following bound on the drift.

Lemma 4.1 (Lyapunov Drift): The one-step Lyapunov drift satisfies the following constraint at any time slot regardless of algorithms to control the system:

$$\begin{aligned}\Delta(\Theta(t)) &= \mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t)) | \Theta(t)\} \\ &\le B + \sum_i \mathbb{E}\{Q_i(t)\lambda_i(t) + Z_i(t)(\rho_i(t) + \lambda'_i(t)) | \Theta(t)\} \\ &\quad - \sum_i \mathbb{E}\{Q_i(t)a_i(t) + Z_i(t)a'_i(t) | \Theta(t)\} \\ &\quad - \sum_i \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)x'_t(t) | \Theta(t)\},\end{aligned} \quad (13)$$

where B is a carefully chosen constant.

By applying the lemma to Eq. (12), we derive the following lemma (some derivations of our lemmas are omitted due to the space limitations).

Lemma 4.2 (Optimization): The minimization problem presented in Eq. (12) can be solved through the maximization problem $\max : \Phi(t)$, where $\Phi(t)$ is defined as follows:

$$\begin{aligned}\Phi(t) = &\sum_i \mathbb{E}\{VU_i^A(t) + \varepsilon(Q_i(t)\mu_i(t) + Z_i(t)\mu'_i(t)) | \Theta(t)\} \\ &+ \sum_i \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)x'_i(t) - VU_i^D(t) | \Theta(t)\}.\end{aligned} \quad (14)$$

B. Worker Selection Algorithm

Based on the above lemma, we propose the following *worker selection* algorithm (Algorithm 1) running at each assignor to maximize $\Phi(t)$.

Algorithm 1: Worker Selection Algorithm for Assignor

```
In time slot $t$ at assignor broker $i$, let $W_i(t)$ be the set of
available workers, then determine 1) worker selection
$\mu_{k,w}(t) = \{0,1\}$ for tasks $k \in Q_i(t)$ and $w \in W_i(t)$; and 2)
task dropping $x_k(t) = \{0,1\}$ for tasks $k \in Q_i(t)$ by solving
the following optimization problem ($r$ as the query of the task):
```

$$\begin{aligned}\max: &\sum_{k \in Q_i(t), w \in W_i(t)} Y_{r,w}^A(t)\mu_{k,w}(t) + \sum_{k \in Q_i(t)} Y_r^D(t)x_k(t) \\ \text{st: } &\mu^{r,w}(t) = \sum_{k \in r} \mu_{k,w}(t) \le 1; \\ &\mu^w(t) = \sum_{k \in Q_i(t)} \mu_{k,w}(t) \le n_w,\end{aligned} \quad (15)$$

$$\text{where} \quad \begin{aligned} Y_{r,w}^A(t) &= VU_{r,w}^A(t) + \varepsilon \cdot Y_r(t); \\ Y_r^D(t) &= [Y_r(t) - VU_r^D(t)]^+; \\ Y_r(t) &= Q_i(t) + Z_i(t) \cdot u_r(t) \cdot e.\end{aligned} \quad (16)$$

Note: since all tasks k of the same query r must have the same $u_k(t)$, here we use $u_r(t)$ in the above equation.

The worker selection algorithm: Algorithm 1 selects (task, worker) pairs $\mu_{k,w}$ and tasks to drop $x_k(t)$ from the queue to maximize the objective function in Eq. (15). The objective function contains two parts: one for worker selection $Y_{r,w}^A(t)$, one for task dropping $Y_r^D(t)$. Both $Y_{r,w}^A(t)$ and $Y_r^D(t)$ take into account the backlogs of queues (i.e., $Q(t)$ and $Z(t)$). Moreover, $Y_{r,w}^A(t)$ takes into account the assignment utility $U_{r,w}^A(t)$, and $U_r^D(t)$ takes into account the dropping utility $U_r^D(t)$. The optimization is subject to two constraints. The first constraint guarantees that the tasks sent to a worker are not the same. The second constraint guarantees that the number of tasks sent to a worker is no more than what he/she asks for.

The optimization problem can be reduced to the *minimum cost maximum flow* problem as we show in Figure 4. In the figure we illustrate an example of the worker selection problem with 6 tasks from 3 queries (query 1 has 2 tasks, query 2 has 3, and query 3 has 1) and 4 workers each of which desires to receive a certain number of tasks. To construct the problem, we create a source node S, a destination node D, a drop node X, and model queries and workers by additional nodes. We form edges from the source node to all queries, edges from a query r to the drop node if $U_r^D(t) > 0$, edges from a query r to a worker w if $Y_{r,w}^A(t) > 0$, edges from all workers to the destination and an edge from the drop node to the destination. We show the (*capacity*, *cost*) pair for each edge in the graph. The capacity from the source to a query is the number of tasks of the query, and its cost is always 0. The capacity from a query to a worker is 1 since only 1 task of the same query is allowed to be assigned to a worker, and its cost is $-Y_{r,w}^A(t)$. The capacity from a query to the drop node is ∞ so there is no limit on the number of tasks can be dropped, and its cost is $-Y_r^D(t)$. The capacity from a worker to the destination is the number of tasks the worker can receive and its cost is always 0. Finally the capacity from the drop node to the destination is ∞ and cost is 0.

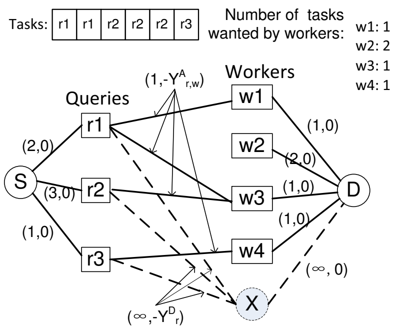


Fig. 4. The representation of a sample optimization problem as a minimum cost maximum flow problem.

By reducing to the minimum cost maximum flow problem we can now use any algorithm for that problem to solve our worker selection problem and it is known that the problem can be optimally solved in polynomial time. One of the well-known algorithms is the *Successive Shortest Path and Capacity Scaling* algorithm by Edmonds and Karp, which is a generalization of the Ford-Fulkerson algorithm [14]. The time complexity of the algorithm is $O(|V|^4 \cdot maxcost)$ where $maxcost$ is the maximum edge cost (relative to 0 as the minimum edge cost).

C. Assignor Selection Algorithm

In this section, we present our design of the assignor selection algorithm, which runs at relays to provide load balancing. Again, we use Lyapunov optimization to design the algorithm, and the beauty of the design is that the system long term goal defined in Eqs. (4a)-(4d) are not compromised by the addition of the algorithm.

We revisit the queuing dynamics of the brokers by taking into account the task flows from relays to assignors. Again, let Q_i denote the task queue at broker i (and the broker could be either an assignor or a relay). $\lambda_i(t)$ is the number of new tasks arriving to broker i. For non-spatial Q&A applications desiring worker selection load balancing, new tasks first arrive at relays for assignor selection. So, $\lambda_i(t) = 0$ for assignors.

Task assignor selection: Every time slot, relay brokers run the assignor selection algorithm to determine assignors for tasks in their queues. Once selected, the tasks are removed from queues at the relay, sent to their selected assignors and put into the queues at the assignors. Let $\omega_{i,j}(t)$ be the number of tasks at relay i selected to assignor j at t. Let $\omega_j(t) = \sum_{i \in Rl} \omega_{i,j}(t)$ be the total number of tasks received by assignor j from relays at t and $\psi_i(t) = \sum_{j \in As} \omega_{i,j}(t)$ be the total number of tasks sent out of relay i at t. Rl and As denote the sets of relays and assignors in the system. Both $\omega_j(t)$ and $\psi_i(t)$ must be finite. We assume they are limited by ω_{max} and ψ_{max} respectively, i.e., $\omega_j(t) \le \omega_{max}$ and $\psi_i(t) \le \psi_{max} \forall t$. The time averaged rates are $\overline{\omega_i} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\omega_i(\tau)\}$ and $\overline{\psi_i} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\psi_i(\tau)\}$.

A generalized queuing dynamics for Q_i at a broker i (indifferent for relays and assignors) is:

$$Q_i(t+1) \triangleq [Q_i(t) + \lambda_i(t) + \omega_i(t) - \psi_i(t) - a_i(t) - x_i(t)]^+, \tag{17}$$

where $\omega_i(t) = a_i(t) = \mu_i(t) = x_i(t) = 0$ if i is a relay, and $\lambda_i(t) = \psi_i(t) = 0$ if i is a assignor.

Relay brokers maintain virtual queue Z_i, the same as we presented earlier. When a task is sent from a relay broker to an assignor broker, its priority-weight is transferred to the virtual queue of the assignor broker from that of the relay broker as well. Therefore, a generalized queuing dynamics for Z_i can be written as:

$$Z_i(t+1) \triangleq [Z_i(t) + \rho_i(t) + \lambda_i'(t) + \omega_i'(t) - \psi_i'(t) - a_i'(t) - x_i'(t)]^+. \tag{18}$$

We update the system Lyapunov function to include Q and Z at relays as well. Using the generalized queuing dynamics in Eqs. (17) and (18), the Lyapunov drift bound is given below.

Lemma 4.3 (Lyapunov Drift): The one-step conditional Lyapunov drift for the entire system of both relays and assignors satisfies the following constraint at any time slot:

$$\begin{aligned}
\Delta(\Theta(t)) &= \mathbb{E}\{L(\Theta(t+1)) - L(\Theta(t)) | \Theta(t)\} \\
&\le B' + \sum_i \mathbb{E}\{Q_i(t)\lambda_i(t) + Z_i(t)(\rho_i(t) + \lambda_i'(t)) | \Theta(t)\} \\
&\quad - \sum_{i \in Rl} \sum_{j \in As} \mathbb{E}\{\nabla_{i,j}^Q(t)\omega_{i,j}(t) + \nabla_{i,j}^Z(t)\omega_{i,j}'(t) | \Theta(t)\} \\
&\quad - \sum_{i \in As} \mathbb{E}\{Q_i(t)a_i(t) + Z_i(t)a_i'(t) | \Theta(t)\} \\
&\quad - \sum_{i \in As} \mathbb{E}\{Q_i(t)x_i(t) + Z_i(t)x_t'(t) | \Theta(t)\},
\end{aligned}$$

where B' is a constant, $\nabla^Q_{i,j}(t) = Q_i(t) - Q_j(t)$ and $\nabla^Z_{i,j}(t) = Z_i(t) - Z_j(t)$.

We consider the same per-slot objective in Eq. (12) and by applying the lemma we derive an additional optimization problem specific for relays besides the original optimization defined in Lemma 4.2.

Lemma 4.4 (Optimization): The minimization problem presented in Eq. (12) can be divided and solved through two maximization problems $\max : \Phi(t)$ and $\max : \Phi'(t)$ where $\Phi(t)$ is defined in Lemma 4.2 and $\Phi'(t)$ is defined as follows:

$$\Phi'(t) = \sum_{i \in Rl} \sum_{j \in As} \mathbb{E}\{\nabla^Q_{i,j}(t)\omega_{i,j}(t) + \nabla^Z_{i,j}(t)\omega'_{i,j}(t)|\Theta(t)\}. \tag{19}$$

It is not hard to see that the above optimization problem for relays is aligned with the optimization for assignors in Lemma 4.2 and they together stabilize the queues and optimize the global long term objective of the system. We propose the *assignor selection* algorithm (Algorithm 2) running at each relay to maximize $\Phi'(t)$.

Algorithm 2: Assignor Selection Algorithm for Relay

```
Let $A_i$ denote the set of candidate assignors for a relay $i$. In
time slot $t$ at relay broker $i$, for each task $k \in Q_i(t)$ select
$j \in A_i$ that the following objective is satisfied:
    max:   $V_k = (Q_i(t) - Q_j(t)) + (Z_i(t) - Z_j(t)) \cdot u_k(t) \cdot e.$   (20)
If $V_k > 0$ then send the task $k$ to $j$. Otherwise, keep $k$ in $Q_i$
and do not send.
```

The assignor selection algorithm: Algorithm 2 takes $O(mn)$ time, where m is the number of candidate assignors of the relay broker and n is the number of tasks in the assignor selection queue. We can see that the selection decisions are made based on the backlogs at assignors. Intuitively, by maximizing Eq. (20) the algorithm achieves load balancing since it prefers assignors with smaller backlogs. Also, if all candidate assignors are loaded (i.e., $V_k < 0$) the tasks are better off held by the relay before they are less loaded.

Last, we omit the performance bounds of our algorithms due to the space limitations.

V. Implementation and Performance Evaluation

In this section, we evaluate our proposed solution using both real implementations and extensive trace-driven simulations.

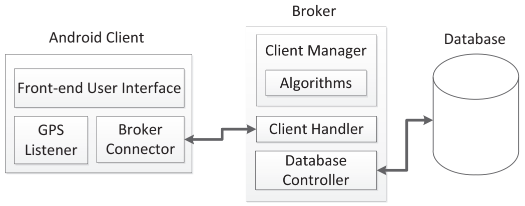
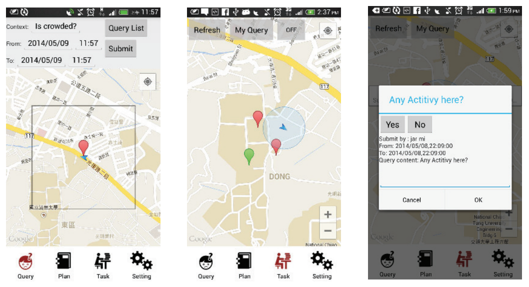


Fig. 5. Prototype architecture.



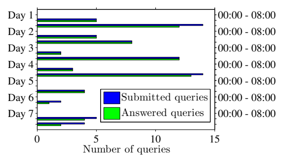
Submit by : jar mi
From: 2014/05/08,22:09:00
To: 2014/05/08,22:09:00
Query content: Any Actitivy here?

(a) (b) (c)

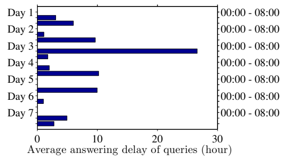
Fig. 6. Screenshots of our prototype: (a) submit a question, (b) display recommended questions, and (c) answer a question.

A. *SmartSource Implementation*

We implemented a prototype system for spatial mobile Q&A and collected real traces from users. The system consists of (i) *broker*, as a Java application deployed on a Linux server and (ii) *client*, as an Android application distributed to mobile users. Figure 5 shows the architecture of our prototype, and Figure 6 shows the screenshots of the client app. Mobile users interact with the app to ask and answer questions. The front-end user interface embeds the Google Map service to allow users to easily target locations for questions (Figure 6(a)). The app provides five predefined questions such as "are the shops open?" and "is crowded?" to allow users to ask questions in a click, or they can type in customized questions. The submitted questions and answers are stored in the database at the server side. Moreover, the client app records the user's location and energy, and periodically reports to the broker. The information is stored in the database as well. The worker selection algorithm is implemented at the broker and recommended questions are shown as markers on the map at the client (Figure 6(b)). Users can click on a marker to answer the question (Figure 6(c)). Users can also check the answers of their own questions.



(a) Number of queries and answered queries.



(b) Average answering delay.

Fig. 7. Prototype usages statistics.

We distributed the client app to 7 students at National Tsing Hua University to use for a week. Some statistic results are shown in Figure 7. Figure 7(a) presents the number of submitted and answered questions for every 8 hours period. The results show that over 80% questions are answered. Also, the peak usage periods are noons and evenings. Figure 7(b) shows the average delay for a question to get answered since it is asked. We observe that most of the time the queries can get the answers in less than 10 hours. This could be a sign that users use the system very frequently since they benefit (receive answers) from the system. We also collect the feedback from the users by questionnaires. The feedback shows that they use the application each day. Most of the users say that the answers are useful to them.

Although we implemented the prototype system to be used in real life, the number of users and questions are still few and cannot provide enough information to understand the performance of the solution. Therefore, we conducted extensive simulations to evaluate the performance of our proposed middleware and algorithms.

B. Performance Evaluation

We conducted several experiments on real-world data. Below we first discuss our trace collection and experimental methodology. We then present our experimental results.

1) Trace Collection: We collected two real-world datasets for the evaluations. The first dataset is obtained from PTT [5], a Taiwan based Bulletin Board System (BBS). It is arguably the largest BBS in the world with more than 1.5 million registered users. We collected 5700 posts in 10 days period from April 11 to April 20, 2014, and treat them as queries. We assume the PTT users as workers. To simulate spatial-temporal Q&A, we extract locations of the posts and users. We assume the location of the author of a post is the requested location of the query. We approximate users' locations from their IPs using triangulation [36], [35]: we recruited three servers in Taiwan and let them ping the IPs to estimate their distances to the servers from the network delay (RTT). We also partitioned Taiwan into grids and determine the location of an IP by computing the Mean-Square-Error (MSE) of each grid's and the IP's distances to servers and finding the minimum. A random offset within a grid is assigned to each user. Moreover, we used random waypoint model to simulate workers movement when they are idle. When workers are performing tasks, they are moving towards the requested locations of the assigned tasks.

To simulate more complicated scenarios, e.g., taking into account the topics of questions in Q&A, we collected the second dataset. We crawled Quora and collected 1188 questions under the topic group "San Francisco" and 1431 users which are answerers of the questions. The topic group contains many sub-topic tags (e.g., "restaurants in San Francisco") and we recorded all the topic tags for the crawled questions and those followed by users, serving as the classes of questions of the queries and interests of the workers. We used the dataset for both spatial-temporal and knowledge-based Q&A systems. To simulate spatial-temporal Q&A system, we imported the mobility trace dataset [31] of approximately 500 taxis collected over 30 days in the San Francisco Bay Area. We randomly assigned the taxis traces to workers and locations to queries. Furthermore, we simulated 3 types of smartphones with distinctive energy costs for sensing topics of questions, and randomly assign smartphones to workers. One the other hand, to simulate knowledge-based Q&A based on social networks [26], we imported the Facebook social graph obtained by McAuley and Leskovec [30], consisting of 193 circles and 4039 users. We randomly map Quora users to Facebook users.

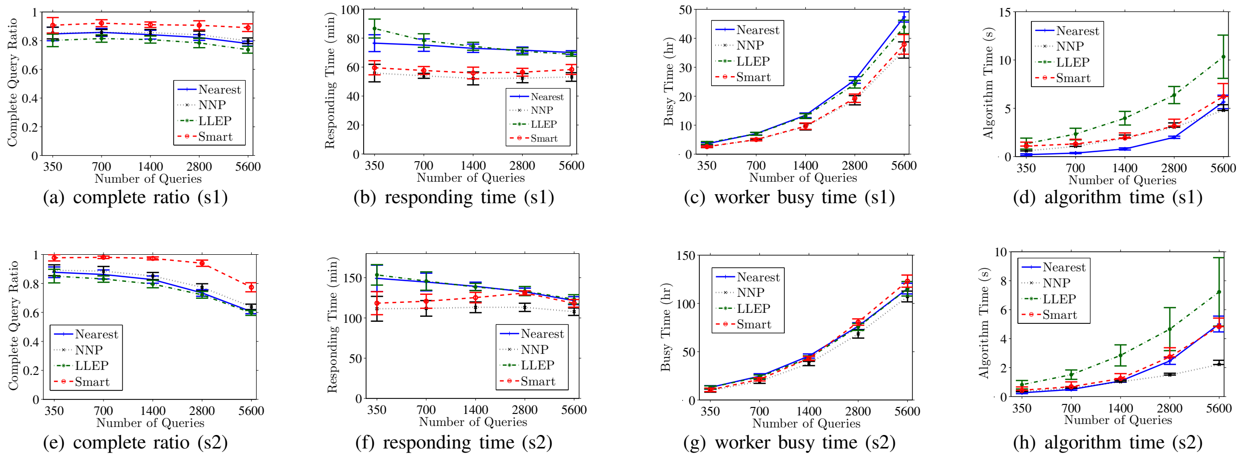
2) Experimental Methodology: We built a Java based simulator driven by the datasets and simulated a Q&A environment of 10 days with time slot of 10 minutes. Each query randomly picks a k_r (i.e., number of required workers) between 1 and k_r^{max} , and has a deadline of 6 hours after posted. Each worker can only accept one task at a time, and will not receive new tasks before he/she finishes the current one. Moreover, workers may not always be available even they have no tasks to perform. We set a probability (50% or 33%) for their availability in each time slot. In our experiments with the PTT dataset, we considered 200 workers and varied the number of queries between 350 and 5600. In experiments with the Quora dataset, we fixed the number of queries to 1000 and varied the number of workers between 100 and 1000.

We considered the following performance metrics: 1) *complete ratio* as the number of queries completed by workers before their deadlines over the total number of queries; 2) *average responding time* as the average time for a query to get answered since it is submitted. For spatial-temporal Q&A the time includes the query queuing time for worker selection, workers' moving time to queried locations, and collecting the queried information. For knowledge-based Q&A, the time is primarily the queuing time and the time to provide answer; 3) *algorithm time* as the computational time of the worker selection algorithms; 4) *worker busy time* and *energy cost* in spatial-temporal Q&A as the total time and total energy cost for workers working on queries; 5) *utility* as the total assign utility (in Eq. (3)).

We conducted three sets of experiments. In the first set of experiments, we evaluated the performance of SmartSource worker selection against baselines for spatial-temporal Q&A. We considered three baseline algorithms for worker selection in each time slot, given a set of spatial-temporal queries and a set of available workers: 1) *Nearest*: a greedy heuristic algorithm that iteratively picks the pair of query and worker that the distance from the worker to the query is the minimum among all query worker pairs; 2) *Nearest Neighbor Priority (NNP)*: an optimal algorithm proposed in [18] to maximize the number of task assignment while minimizing the travel cost of the worker to move to the requested location; 3) *Least Location Entropy Priority (LLEP)*: an algorithm proposed in [18] to assign higher priority to tasks which are located in worker-sparse areas. In the second and third sets of experiments, we evaluated the implication of system parameters on the performance of SmartSource in the spatial-temporal and knowledge-based Q&A systems, respectively.

3) Experimental Results: The experimental results are given below.

Compared with other strategies. Figure 8 shows the comparisons of SmartSource against baseline algorithms on spatial-temporal Q&A. We used the PTT dataset and varied the number of queries submitted by querists. Since the baseline algorithms perform worker selection only based on the locations



(a) complete ratio (s1) (b) responding time (s1) (c) worker busy time (s1) (d) algorithm time (s1)

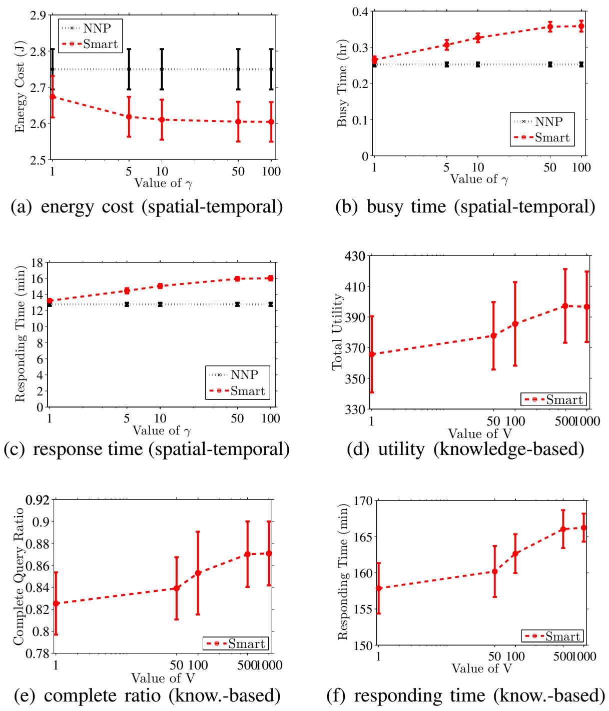
(e) complete ratio (s2) (f) responding time (s2) (g) worker busy time (s2) (h) algorithm time (s2)

Fig. 8. Comparing SmartSource with others on spatial-temporal Q&A, s1: $k_r^{max} = 3$, 50% worker availability; s2: $k_r^{max} = 6$, 33% worker availability.

of queries and workers, for fairness in comparisons we adjust the utility in SmartSource to consider moving cost only. We simulated two scenarios $s1$ and $s2$ with different worker demands from queries and worker availability. Compared to $s1$, $s2$ represents scarcer worker resources due to higher worker demands and lower worker availability. We observe that with the increase of the number of queries, the complete ratio of all algorithms are decreasing due to the lack of worker resources. Compared with other strategies, SmartSource achieves 90% complete ratio, which is over 10% higher than other algorithms in $s1$ (Figure 8(a)) and it is less impacted by the increase in the number of queries. In $s2$ where the worker resources are scarcer, SmartSource is able to maintain 80% complete ratio, which is over 20% higher than baselines (Figure 8(e)). This owes to its deadline-aware queuing and worker selection mechanism.

We also evaluated the query responding time and worker busy time as an indication of how fast can queries be answered and how much work load for workers. We observe that in spatial-temporal Q&A system, workers' travel time to requested locations is an important factor impacting the query responding time and worker busy time. NNP has lowest query responding time (Figures 8(b) and 8(f)) and worker busy time (Figures 8(c) and 8(g)) because it assigns tasks to workers with minimum traveling distance. SmartSource also minimizes workers' traveling cost but has a longer query responding time than NNP primarily because of the queuing delay of the queries before they are assigned. With the increase of number of queries, the total worker busy time (Figure 8(g)) is increasing for all algorithms because more tasks were performed. We observe that although SmartSource workers performed more tasks than those in other strategies (because of a higher query complete ratio), their total busy time are close.

Figures 8(d) and 8(h) show the total computational time of the algorithms for worker selection. We can see that the computational time of all algorithms are increasing with the increase of the number of queries. SmartSource has a similar computational time compared to other strategies but achieves better worker selection performance.



(a) energy cost (spatial-temporal) (b) busy time (spatial-temporal)

(c) response time (spatial-temporal) (d) utility (knowledge-based)

(e) complete ratio (know.-based) (f) responding time (know.-based)

Fig. 9. Performance tradeoff of SmartSource by varying γ in spatial-temporal Q&A and varying V in knowledge-based Q&A.

Performance tradeoffs of the algorithm. We evaluated the performance of the SmartSource worker selection under varying system parameters. We used the Quara dataset to take into account question topics, sensing energies and social networks in the simulations. We first evaluated the tuning of parameters of the utility function. Figures 9(a)–9(c) presents its performance by varying γ (i.e., weight for energy cost) in spatial-temporal sensing applications. We used the performance of NNP as a baseline. We observe that with the increase of γ the sensing energy cost of workers in SmartSource are decreasing (Figure 9(a)). However, this is accompanied with an increase in worker busy time (Figure 9(b)) and query response time (Figure 9(c)). This is because the worker selection places higher priority on minimizing workers' energy cost for sensing by compromising workers' traveling distance, which leads to an increase in the overall working time on spatial queries. This demonstrates the flexibility of our middleware: it can be customized for different application by tuning parameters in the utility function.

We further evaluated the performance of the SmartSource by varying V (i.e., weight for utility over queue length). We considered a knowledge-based Q&A and set $\beta = \gamma = 0$ in the utility so the worker selection mainly considers the interest similarity and social closeness. Figures 9(d)–9(f) present the results. We observe that with larger V, SmartSource is able to achieve higher utility (Figure 9(d)) and higher query complete ratio (Figure 9(e)). However, although larger V leads to higher utility, it also leads to larger queue size and longer queuing delay. This can be seen from the increase in the query response time (Figure 9(f)). Moreover, the total utility becomes stable while V keeps growing, because it is approaching the long term optimal utility. Q&A system developers may select the most suitable V value to meet their design goals.

VI. SmartSource's Application Scenarios

Our proposed SmartSource middleware may be used to develop various mobile Q&A systems. We present two sample application scenarios below.

Spatial-temporal mobile Q&A systems allow mobile users to ask/answer spatial-temporal dependent questions [12], [28], [27], [18], [11], [19], [33]. For example, Deng et al. [12] study the problem of the worker selection, in which each incoming worker is given a schedule of requests to answer, and the goal is to maximize the number of completed requests. The algorithms are proposed, based on dynamic programming and branch-and-bound. Kazemi and Shahabi [18] study the problem of simultaneously determining the schedules for multiple workers, and convert the problem into a maximum flow problem with an objective of maximizing the total number of assigned requests. Liao and Hsu [28], [27] study the problem of spatial-temporal dependent multimedia sensing tasks, where photo/video of a target may be taken from multiple spots, which incurs even higher complexity than the problem considered in Deng et al. [12]. Liao and Hsu convert the crowdsourcing problem into a variation of orienteering problems [34], and propose several algorithms, including an optimal algorithm [28] and an approximation algorithm [27] to solve the problem. Dang et al. [11] consider a similar problem but with task dependency, and Kazemi et al. [19] consider the problem of fusing inputs from multiple mobile users using majority vote. Tamilin et al. [33] develop a smartphone based crowdsourcing system. Their system selectively presents the queries that match the workers' contexts the most, while achieving good energy efficiency.

Knowledge-based mobile Q&A systems have already commercialized. Mobile Q&A services like Naver Mobile Q&A and ChaCha made it easy for mobile users to find information by asking questions via instant messaging or SMS. Moreover, with the popularity of smartphones, traditional web based Q&A systems such as Answer.com, Quora and Baidu Zhidao have published their mobile apps as extension of services. Several research projects have studied knowledge-based mobile Q&A as well. Lee et al. [23] studied the usage of mobile Q&A by analyzing questions and answers from Naver Mobile Q&A and a survey study of mobile users. They found that the questions asked through mobile Q&A significantly differs from that of web based Q&A as they are deeply wired into users' everyday life activities. In another work, Lee et al. [24] analyzed mobile pay-for-answer Q&A services and they found that in these applications workers are mainly motivated by financial incentives and intrinsic factors (e.g., fun and learning) rather than social factors, and answers are provided quickly. Shen et al. [26] proposed a fully distributed mobile Q&A system based on social networks. The system help mobile users to determine potential answerers in their friend lists to forward their questions to, based on the their interests and social closeness.

VII. Concluding Remarks

We expect that mobile Q&A applications will be a dominant mode of information exchange in future systems since they pave the way for combining the capabilities of powerful technologies in existence today – mobile platforms, social networks and knowledge bases. Such a Q&A paradigm for information exchange opens up a new set of research issues. The ability to support personalized and reliable Q&A in unreliable settings is particularly interesting - a particular use case would be one of exploiting the crowd to gather and distribute situational awareness in disasters. Today, several notification systems are in use and being developed to relay critical content to officials and citizens in harm's way using dashboards, subscription based notification applications, and etc. [6], [7], [22], [16]. Notification is usually triggered by the publication of a message; a more interactive paradigm is necessary when individuals seek specific information. The above Q&A paradigm enables this higher level of interactivity that is driven by the querier/seeker of information. The ability to provide seamless exchange in real-time and in the presence of faults is another huge challenge. Techniques to exploit the underlying network structure [6] may be extended to enhance the reliability of exchange using resilient overlay networks. Finally, human-oriented Q&A systems are extremely tolerant to inaccuracies in formulation of questions/answers as opposed to search-based systems. For example, a querist asking for the "crowd-level at the campus medical facility" can express the needs loosely; a responder might indicate "I have been waiting in line for an hour" - such loosely formed questions/answers are still useful in this paradigm. Translating this flexibility into the needed quality of queries/answers at lower levels is interesting and can be used to reduce system overheads. For example, the authors of [7] proposes a service for flexible localization to satisfy the diverse localization quality levels required by different applications; studies in the literature [22], [16] consider the impact of uncertainty of the sources to ensure the answers are acceptable.

References

[1] Amazon mechanical turk. http://www.mturk.com.

[2] Crowdflower. http://www.crowdflower.com.

[3] Google map service. https://developers.google.com/maps/documentation/webservices/.

[4] Mobile subscribers ages 55+ own smartphones. http://tinyurl.com/mg5glzw.

[5] Ptt bbs. http://www.ptt.cc/index.html.

[6] K. Benson and N. Venkatasubramanian. Improving sensor data delivery during disaster scenarios with resilient overlay networks. In *Proc. of PERCOM'13 Workshops*, 2013.

[7] S. Bonetti, S. Mehrotra, and N. Venkatasubramanian. Exploring quality in multisensor pervasive systems - a localization case study. In *Proc. of PERCOM'10 Workshops*, 2010.

[8] Y. Chon, N. Lane, Y. Kim, F. Zhao, and H. Cha. A large-scale study of mobile crowdsourcing with smartphones for urban sensing applications. In *Proc. of UbiComp'13*, 2013.

[9] Y. Chon, N. Lane, F. Li, H. Cha, and F. Zhao. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proc. of UbiComp'12*, 2012.

[10] V. Coric and M. Gruteser. Crowdsensing maps of on-street parking spaces. In *Proc. of DCOSS'13*, 2013.

[11] H. Dang, T. Nguyen, and H. To. Maximum complex task assignment: Towards tasks correlation in spatial crowdsourcing. In *Proc. of iiWAS'13*, 2012.

[12] D. Deng, C. Shahabi, and U. Demiryurek. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In *Proc. of SIGSPATIAL'13*, 2013.

[13] N. Do, Y. Zhao, S. Wang, C. Hsu, and N. Venkatasubramanian. Optimizing offline access to social network content on mobile devices. In *Proc. of INFOCOM'14*, 2014.

[14] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 1972.

[15] L. Georgiadis, M. Neely, and L. Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, 2006.

[16] Q. Han and N. Venkatasubramanian. Information collection services for qos-aware mobile applications. *IEEE Transactions on Mobile Computing*, 2006.

[17] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele. Participatory air pollution monitoring using smartphones. In *Proc. of International Workshop on Mobile Sensing*, 2012.

[18] L. Kazemi and C. Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *Proc. of SIGSPATIAL'12*, 2012.

[19] L. Kazemi, C. Shahabi, and L. Chen. GeoTruCrowd: Trustworthy query answering with spatial crowdsourcing. In *Proc. of SIGSPATIAL'13*, 2013.

[20] K. Kim, Y. Zhao, and N. Venkatasubramanian. Gsford: Towards a reliable geo-social notification system. In *Proc. of SRDS'12*, 2012.

[21] N. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha. Piggyback crowdsensing (pcs): energy efficient crowdsourcing of mobile sensor data by exploiting smartphone app opportunities. In *Proc. of SenSys'13*, 2013.

[22] I. Lazaridis, Q. Han, S. Mehrotra, and N. Venkatasubramanian. Fault-tolerant queries over sensor data. In *Proc. of COMAD'06*, Delhi, India, 2006.

[23] U. Lee, H. Kang, E. Yi, M. Yi, and J. Kantola. Understanding mobile q&a usage: an exploratory study. In *Proc. of CHI'12*, 2012.

[24] U. Lee, J. Kim, E. Yi, J. Sung, and M. Gerla. Analyzing crowd workers in mobile pay-for-answer q&a. In *Proc. of CHI'13*, 2013.

[25] Z. Li and H. Shen. Soap: A social network aided personalized and effective spam filter to clean your e-mail box. In *Proc. of INFOCOM'11*, 2011.

[26] Z. Li, H. Shen, G. Liu, and J. Li. Sos: A distributed mobile qa system based on social networks. In *Proc. of ICDCS'12*, 2012.

[27] C. Liao and C. Hsu. An approximation algorithm of orienteering problems for mobile computing. In *Proc. of MobiSys'13*, 2013.

[28] C. Liao and C. Hsu. A detour planning algorithm in crowdsourcing systems for multimedia content gathering. In *Proc. of MoVid'13*, 2013.

[29] L. Massoulie, E. Merrer, A. Kermarrec, and A. Ganesh. Peer counting and sampling in overlay networks: random walk methods. In *Proc. of PODC'06*, 2006.

[30] J. McAuley and J. Leskovec. Learning to discover social circles in ego networks. In *Proc. of NIPS'12*, 2012.

[31] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.org/epfl/mobility/, 2009.

[32] M. Talasila, R. Curtmola, and C. Borcea. Improving location reliability in crowd sensed data with minimal efforts. In *Proc. of WMNC'13*, 2013.

[33] A. Tamilin, I. Carreras, E. Ssebaggala, and A. Opira. Context-aware mobile crowdsourcing. In *Proc. of UbiComp'12*, 2012.

[34] P. Vansteenwegena, W. Souffriaua, and D. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 2011.

[35] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang. Towards street-level client-independent ip geolocation. In *Proc. of NSDI'11*, 2011.

[36] B. Wong, I. Stoyanov, and E. Sirer. Octant: A comprhensive framework for the geolocalization of internet hosts. In *Proc. of NSDI'07*, 2007.

[37] T. Yan, V. Kumar, and D. Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *Proc.of MobiSys'10*, 2010.

[38] P. Zhou, Y. Zheng, and M. Li. How long to wait?: Predicting bus arrival time with mobile phone based participatory sensing. In *Proc. of MobiSys'12*, 2012.