MediaServ: Subscription-based Media Crowdsourcing for Collective Events

A s m Rizvi, Shamir Ahmed, Minhajul Bashir, and Md. Yusuf Sarwar Uddin
Department of CSE, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
Email: {rizvi22222, shamir253, shovan.livebd}@yahoo.com, yusufsarwar@cse.buet.ac.bd

***Abstract*—In this paper we propose resource optimization for subscription based media content crowdsourcing. In this form of crowdsourcing, interested entities (referred to as Campaigners) announce their interests expressing what media content they want to receive from users whereas users subscribe to those interests to serve content satisfying the interests. Campaigners solicit content generated by users by mentioning criteria that the media content should satisfy, for example a noise pollution campaigner who wants to measure noise level of a city, may ask for audio clips recorded at a certain location. Subscribed users voluntarily or on paid terms generate content against those interests. Given that a user may subscribe to different campaign interests, its generated content may satisfy different interests in varying degree of accuracy, we propose methods to evaluate contents based on the degree of satisfaction against the interests, and develop techniques for delivering those contents.**

I. Introduction

In this paper, we propose a media data collection system that collects media content from citizens against expressed subscriptions for collective events. We refer collective events as a set of social or citizen assisted events that require content contribution from participants. This paper proposes content upload policies for subscription based media crowdsourcing (SMCS). SMCS refers to a participatory platform where interested parties can create events and ask ordinary citizens to voluntarily deliver a certain type of media contents to a service point of the event generator. Example may include a City authority asking its citizens to report problems of inhabitants by taking *photos*, such as an open sanitary leak. Department of environment may ask citizens to capture audio clips for reporting noise level, or transportation department may ask for pollution incidents captured by videos of vehicles releasing black smokes. This type of service modality in emerging due to two reasons. One is the abundance of mobile phones equipped with high resolution camera, audio/video capture capabilities possessed by average individuals (study predicts the number of phones may exceed 2 billion in 2015 [1]). The second reason is the generation and distribution of huge amount of media contents through applications, such as Instagram, Flickr (photo sharing), YouTube, Vimeo (video sharing), and SoundCloud (audio sharing). Record shows nearly 350 million photos are uploaded daily on Facebook [2]. This paper leverages these opportunities to address a question whether we can channelize these contents in more targeted end points by explicit intents and generate values from the content uploaded. The initial idea of this paper was also appeared in [3]. In this paper, we elaborate the initial idea with simulation and real experiment results.

Subscription based media crowdsourcing is yet another publish/subscribe system, a set of requesters, we call them *campaigners*, solicit content from users by mentioning explicit *interests* of certain media content. These requests are referred to as *campaigns*, which are described by a set of conditions that the requested contents should satisfy (e.g., pictures taken from a certain location during daytime). These citizen assisted events need large amount of media contents from participants of users. Interested mobile users may register to these campaigns and publish content. Unlike traditional pub/sub where mobile user are mostly treated as subscribers receiving published contents, here in our context they are rather publishers, uploading contents for campaigns. As users may subscribe to several such campaigns or events, an issue becomes important how much resource the user may want to engage in to serve them. Admittedly media contents are large data objects so it usually takes considerable amount of resources to process and upload them. In this paper, we address the problem of content uploading in the context of subscription based media crowdsourcing for collective events.

Since mobile users rely on metered 3G or 4G connections that provide only limited data usage (as they are charged heavily otherwise), data transfer capacity becomes one of the sole constraints; more specifically the budget for content uploads to the service end points. Our formulation takes this into consideration. As user generated contents may satisfy different subscribed interests in varying degree of accuracy, we propose methods to evaluate those contents based on the degree of matching against the subscribed interests. Again, different services could have different level of importance (may be set forth by users). Considering all these, we develop a formulation for delivering contents that optimizes the user's resource utilization.

To this end, we propose *MediaServ*, a system that handles subscription based media crowdsourcing for a mobile device and optimally allocates resources for multiple such subscriptions. Even though the proposed system is quite general in terms of types of media content it can serve (such as image, audio), we explicitly limit our work and experiments on pictures, mainly because pictures are predominant media content to date, and their use-cases are most befitting to our design. In that, we assume users subscribe to interests that ask for pictures. Our system has an Android based app that manages subscriptions, captures and uploads images against registered campaigns respecting individual data budget constraint.

II. Related Works

Some photo retrieval service also exists. One example is MediaScope [4] , which tries to fill up the availability gap.

978-1-5090-1269-5/16/$31.00 ©2016 IEEE

Availability gap means the time between which the item is taken and when it is shared (uploaded to a sharing site). On the other hand, a subscription based media crowd-sourcing which targets to optimize resource consumption is proposed by us. In our system user evolution like giving caption, selecting category is counted. Photo recognition in crowd sourcing based multimedia by the users gives almost correct value in different environments [5]. Some picture delivery service is also available. One example is geo-pictures (http://www.geo-pictures.eu/), which relies on satellite communication for sending pictures to the command station. For disaster recovery, there is also PhotoNet service [6]. Different user behavior analysis based on uploaded photos in Instagram is demonstrated in [7].

Mobile participatory sensing has also been used for capturing sound data. One such system has been proposed in [8], which focuses on measuring noise pollution levels. Another system, NoiseSPY [9], monitors noise to measure and anotate urban noise using smartphone sensing. NoiseTube [10] is another platform that utilizes noise data from smartphones as well as manual contextual data like location, time and noise source, and data generated by machine-based automatic classifiers.

In mobile participatory sensing, security and large user participation are the most common challenges. Many researches have been conducted regarding these two problems. Some incentive mechanism has been designed for mobile sensing in [11]–[13]. A recurrent reverse auction incentive mechanism with a greedy algorithm has been discussed in [11]. Based on Stackelberg game, and auction-based incentive mechanism a procedure is discussed in [12]. Efficient bidding system for user is discussed in [13]. Security issues are discussed in [14]–[16]. Framework Pepsi is discussed in [14]. Encrypted image based solution is given in [16]. Privacy-aware incentive mechanism is discussed in [17].

III. Content Upload in Subscription Based Media Crowdsourcing

Subscription based media crowdsourcing has three components: *campaigners*, *interests*, and *publishers*. Campaigners are individual persons or organizations who are in need of certain media content. Campaigners initiate a campaign by registering to the system by specifying their interests. These interests are simple predicates that media objects should satisfy (e.g., pictures from location X). Mobile users, act as publishers in the system, register to the campaigns they are interested to publish contents against. This registration is also known as *subscription*. Note that the term 'subscription' is meant different than its traditional meaning. Here subscription does not refer to receiving contents, rather publishing contents. When a content gets generated at a user (e.g., user takes a picture), the content is matched against all subscriptions whether it satisfies any interest of the subscribed campaigns of that user. Content is uploaded to the backend server only if it matches any subscription. Figure 1 shows the interaction.

While the development of entire pub/sub system is possible using existing pub/sub middleware, the full description with detail technical merit is beyond the scope of this paper. This paper focuses mainly on the operations that mobile publishers perform when they generate media contents and upload them to the system. We indeed do build a prototype system that entails all components, but may be in smaller scale, only enough to evaluate data publishing part of the system. Two important aspects are considered in this connection. One is the automatic scheduling of content uploads based on the subscriptions held by the user. The second one is related to specifying user's data budget constraint on its content uploads. In the following, we describe these two issues.

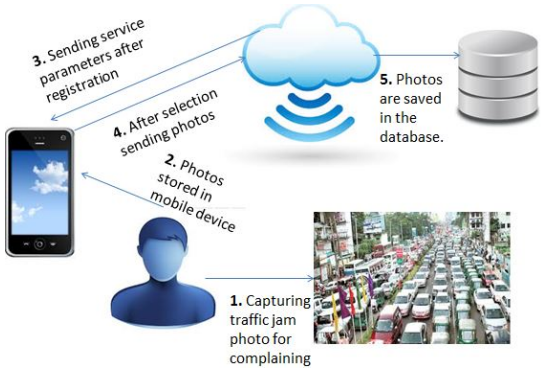


Fig. 1: System component and flow

A. Formulation of Content Uploads

When mobile users generate contents against their subscriptions, the contents are not uploaded immediately; instead they are uploaded in a *batch*, once in every so often defined by a *period*. This is for two reasons. Contents may arrive at a high rate in which immediate uploads may cause high processing at the mobile device. Again, the device may not be connected to the Internet allowing uploads not to happen anyway. We refer to these time instances as *upload events* when uploads can happen. If the user's device is offline at the event time, the interval is stretched until the next connectivity occurs. Contents are queued between events.

Let $c_1, c_2, \cdots, c_n$ be the set of contents (indexed by i) at an upload event, and $s_1, s_2, \cdots, s_k$ be campaigns (indexed by j) that the user subscribed to. For each content, user may choose one or more campaigns where that content belong to. We define a matching score, $\psi(i,j)$, that measures to what extend content i *satisfies* the requirement set forth by campaign j. This is measured from media metadata and the requirement parameters set by the campaigners. Without of loss generality, we can assume $0 \leq \psi(i,j) \leq 1$; where $\psi(i,j) = 0$, if content c_i is not assigned for s_j or it does not satisfy campaign requirement at all, and $\psi(i,j)$ becomes 1 if it satisfies the interest completely. Any real number between 0 and 1 designates the degree of matching that content i demonstrates against the campaign requirement.

Campaigns may have differentiated importance or priorities overs others. These priorities can be set by the service providers when they register to the system or, alternatively, can be set by the user itself to differentiate among its subscribed campaigns. Priorities, denoted by $p(j)$, are real numbers ≥ 1, higher value means higher priority. Based on campaign priorities, we can define a *value* for each content as the weighted sum of all matching scores across all campaigns the

content satisfies. That is, $value(i) = \sum_j p(j)\psi(i,j)$. Note that $value(i) = 0$ means content does satisfy none.

Let x_i be a binary decision variable that indicates whether the content i should be uploaded at the current upload opportunity. Obviously, we discard items with $value(i) = 0$. Moreover, a user cannot upload all candidate objects at upload opportunities because it might have a data upload budget. We assume that users' devices are on metered cellular connections and that users set explicit upload budget based on their connection plans. So, we want to upload contents so as to—

$$\max \sum_i x_i \times value(i), \text{ subject to } \sum_i x_i \times size(i) \leq B$$

where $size(i)$ denotes the size (in byte) of content i, and B is the total transfer volume allocated for at that upload event. Budget, B, restricts how many bytes can be uploaded at this upload instance. The formulation is an instance of the classical 0/1 Knapsack problem which is reportedly NP-Hard. We use greedy heuristics solving for x_i, particularly, pick items in descending order of $value(i)/size(i)$ until the budget is full.

A few modifications can to the formulation is possible. We can limit low matching contents not to upload when their score falls below a threshold. The budget constraint can be written in various other forms too, such as

$$\begin{aligned} \text{Budget per campaign} \quad & \textstyle\sum_i x_i \times sz(i) \leq B(j), \forall j \\ \text{Weighted budget} \quad & \textstyle\sum_i x_{ij} \times sz(i) \leq \left(\frac{p(j)}{\sum_l p(l)}\right) \times B, \forall j \end{aligned}$$

What remains to discuss is how to design matching function, $\psi(i,j)$, for contents against campaigns. We do not tie this strongly in our service rather leave it open for campaigners to specify. For example, some campaigns may describe their interests by specifying explicit *predicates* over a set of attributes requesting items within a certain range of their attributes values (i.e, pictures from an area), which precisely defines the matching criteria. Another set of campaigns may specify a *reference* deviation from which measures degree of matching (i.e., pictures near city center (lat, lng) on Oct 31, 2015). In our system, we use spatio-temporal attributes, namely date, time, and location as reference attributes for requests, and define a distance function between a content and a campaign as follows:

$$dist(i,j) = w_1 dist_t(i,j) + w_2 dist_l(i,j)$$

where $dist_t$ and $dist_l$ measure much content i deviated in time and location for the reference point of campaign j. Weights w_1 and w_2 define sensitivity to these two distance components. Afterwards, we simply use $\psi(i,j) = e^{-D(i,j)}$ as the matching score. We assume each campaign specifies a spatio-temporal reference for their interests with an associated weight values.

B. Data Budget Constraints for Mobile Users

In our system mobile users use cellular data connections with limited data plan, which does not allow them to use all of their data plan for a single application. So, it is natural that they put a budget on their data activity per application. Therefore, our MediaServ app receives a budget, which is specified by a tuple (D,T), indicating that the app is allowed to upload D amount of bytes for every T unit of time (we refer to as *round*). For simplicity, we assume T is an integral multiple of our scheduling period, P. After T amount of time elapses, the user receives another chunk of D bytes to replenish its remaining budget for the successive operations. We assume that remaining budget rolls over from one round to the next.

MediaServ uses two budget allocation techniques to determine how much data can be uploaded at each upload events. Recall that upload events are at least P time apart. In the first scheme, the app assumes all D bytes are available from the beginning and uses up the budget as upload happens until it hits zero. This scheme is called *gradual decrease*. In the other scheme, budget is released rather slowly at a constant rate $R = \frac{D}{T}$. That means, full D bytes are not available at the beginning; rather accumulates as time passes. This technique is referred to as *gradual release*. In this case, the application maintains a running budget counter, B, to hold much it is allowed to spend now, which is updated as follows: $B_{now} = B_{old} + R(t_{now} - t_{last})$, where t_{now} means current time and t_{last} is the last time a budget was replenished. Even though both schemes are confirmed to use the same amount of uploads, we see in experiments that graduate release performs better than gradual decrease.

We also propose another budget allocation scheme that lies in between of the above two. This uses a *varying* replenishment rate in contrast to constant rate as in gradual release. The gradual release scheme releases takes the full duration, T, to release the entire budget at a constant rate, $r(t) = R$. In varying rate case, we allow entire budget to get released by βT time for a control parameter $\beta \leq 1$. That means, at the beginning the process has a higher rate of release, and then the rate slows linearly down to zero after βT. We can visualize the rate line, $r(t)$, along time, t, to form a right triangle whose base in βT, and the height is such that the entire area bounded by the triangle is D (i.e., total budget). So, the height is $\frac{2D}{\beta T} = \frac{2R}{\beta}$, which gives the rate equation as follows:

$$r(t) = \frac{2R}{\beta}\left(1 - \frac{t}{\beta T}\right) \tag{1}$$

Hence, update rule for B for this scheme becomes:

$$B_{new} = B_{old} + \int_{t_{last}}^{t_{now}} r(t)dt$$

C. Scheduling for Multiple Campaigns

When a user subscribes to multiple campaigns, the formulation mentioned in Section III-A favors campaigns with higher priority, which may cause an unfair distribution of resources across campaigns. To mitigate this, in addition to greedy heuristic mentioned earlier, we propose a few heuristics that addresses usage of upload resources at varying degree of fairness.

First obvious choice could be *FIFO* order, contents are uploaded in temporal order. This requires little processing on content collection as it does not need further processing other than maintaining a queue. Another approach can be *round robin*; one content from each campaign at a time and continuing. In this, all campaigns are treated alike, and their

priority values are ignored. Contents under each campaign however sorted in descending oder of their *value-size* ratios. This perhaps gives the most fair usage of upload opportunities across campaigns. A variant of round robin can be *probabilistic* method where campaigns are chosen randomly based on the weighted priority. In that, the probability that campaign j would be chosen next is equal to $\frac{p(j)}{\sum_j p(j)}$. The final one can be strictly in order of campaign priority. Contents from the top campaigns are all uploaded first then moving to the next important campaign. This is useful in contexts where campaigns are very short-lived and required contents all in a certain time deadline (for example, journalists cover stories for tomorrow's daily).

IV. Simulation Results

We do a simulation based study to demonstrate our budget scheme on content uploads. We build a custom discrete event simulation using Java to simulation a mobile client with varying network connectivity (the device ON/OFF pattern is modeled using a Poisson process). We use a Poisson arrival process of media content generation with a rate of 1 per thirty minutes. Contents are images of random size 0.5MB to 2.5MB. We use varying degree of data budgets for a duration of 15 days.

To introduce variability in content generation and their impact on uploads against a given budget allocation scheme, we use three more data generation scenarios in addition to a constant rate. These incorporate various rates at different interval of the simulation. First one generates content at a high rate at the beginning during first half followed by average rate at the second half. The next is the reverse: low followed by high. The last one alternates between high and low for a quarter duration of each. Table I shows the simulation environment.

TABLE I: Simulation Environment

Parameter	Value
Content value(priority)	1-10
Content size	0.5 MB to 2.5 MB
Content generation rate	1 per 30 minutes (Five times faster at high rate)
Network on	Average 10 minutes
Network off	Average 30 minutes
Simulation run for	15 days
Net speed	50 kbps
In gradual budget	Increase by a constant(budget/days) after every day
Total budget	50 MB to 400 MB
Seed values	2000 to 2012

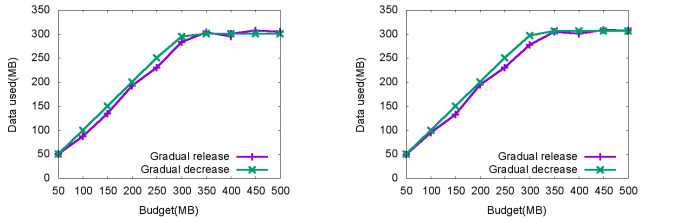
In all cases, we are interested in different performance metrics, such as number of content uploaded, total value of the all sent contents, total bytes sent, and number of dropped contents. Due to space limit, we here produce results on total content uploaded, referring to [18] for detailed results. In the following, we show results for a single subscription, leaving the same for multiple subscriptions (for full results, see [18]).

In Figure 2, we show comparison between two budgeting schemes, gradual release and decrease, at different content generation rates. It is apparent from all four graphs that, budget less than 250MB, gradual release scheme shows better result. After 250MB, gradual decrease shows similar or better results. Before 250MB, resource is scarce and we need to select which contents we choose to upload. But after 250MB, of total budget we have a large amount of budget and we can use it without much concerning about our limitations. We may find an intersection point where two schemes have the same results. After that, most of the cases, gradual decrease method shows good result. In Figure 2c and Figure 2d, the difference between the two graphs is more prominent.

When we wait for contents with higher values and delay uploads, the content list gets bigger as contents are generated during the wait time. When we get new allocation, may be we wasted a lot of time in waiting, hence we can not transfer as many contents as we can with gradual decrease method. This is the case when budget is larger than around 250MB. But when budget is very low, chance is that we send bigger sized contents too much and wasted a lot of bandwidth. That is why from all the figures, we can see for lower amount of budget gradual release method shows better result.

In Figure 2b, we see that there is no significant difference between two curves. This is the case when content generation rate is very high at the beginning. Hence, there are many contents which were created during the high generation rate period. Gradual decrease method starts to send the contents immediately but gradual release method delays upload because it has low budget at the beginning. As the content list grows, the scheme gets the chance to find smaller sized contents among many contents. Hence, in this scenario it sends somehow more smaller sized contents than the other generation rate scenario. If most of the contents are generated within a short interval, then the gradual decrease method gives better result and the transmission of the contents will be much faster than the gradual release method.

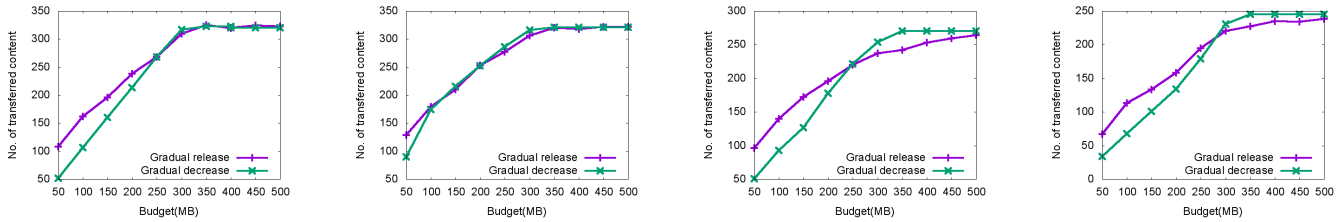
Even though both schemes send varying number of contents, they effectively upload almost the same amount of bytes (see Figure 3a and Figure 3b), which indicates that graduate release uploaded more smaller sized objects than large sized one.



(a) Constant rate (b) High rate followed by low rate

Fig. 3: Comparison between budgeting methods at different rate of content generation

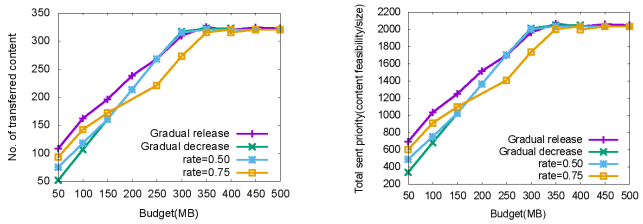
Budget can be released in a different rate rather than a constant rate. The release rate may be higher than the constant rate. As a result all the budget will be released earlier, before the end of the simulation. We simulated for two different rates. A rate indicates the fraction of total time, that is necessary to release all the budget. In Figure 5a and Figure 5b we see the comparison. At the beginning we have two lines in between gradual release and gradual decrease lines. When rate is 0.5 the line follows the line of gradual decrease method. This is because the whole budget is released within $\frac{T}{2}$ time. This is



(a) Constant rate (b) High rate followed by low rate (c) High low alternates (d) Low rate followed by high rate

Fig. 2: Comparison between budgeting methods at different rates of content generation

some-what likely with gradual decrease method.



(a) No. of transferred content comparison (b) Total sent value(priority) comparison

Fig. 4: Comparison among budgeting methods

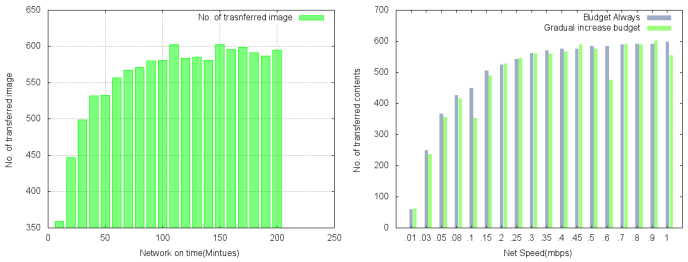
One more thing, we can see in all graphs that the lines are capped after around 250MB budget. As we generate 1 content per 30 minutes, there will be around 720 contents in fifteen days. But before transferring all of these and fills up the whole budget the 15 days time of simulation gets finished. That is why the lines are capped after 250MB of budget.

We can conclude that gradual release budget scheme in general is better than its counterpart. Actually, the performance is largely dependent on content generation rate and their sizes. We observe that when the budget is small following gradual release method is good. When the budget is large gradual decrease shows some promise for better results.

We also did experiments with varying mean network ON time and varying data transfer rate (defaults are average 10 minutes and 50 kbps respectively). Network ON time indicates on an average how long the cellular connection remains once it gets connectet. Figure 5 shows results. We see that as we increase network ON time and transfer bandwidth more and more pictures are uploaded.

V. Implementation

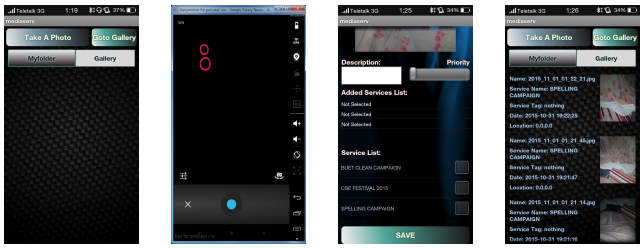
We build an Android app of MediaServ that allows user to create campaigns and publish photos to campaigns. Figure 6 shows the various screens of the app. At first, the user gets four options to select from (Figure 6a). When users subscribe to a campaign, the device is registerred for that campaign and the user is now can publish content for that campaign. When a user intents to publish content, the system camera is invoked



(a) Varying network ON time (b) Varying transfer bandwidth

Fig. 5: Uploads for varying network ON time and varying transfer bandwidth.

to capture photo of the target. After capturing the photo, user can choose to which campaign it wants to publish this photo. Users can also select pictures from its gallery for campaigns. Once captured or selected, pictures are queued internally to be scheduled in the next upload opportunity. A background Android Service maintains this list and uploads pictures from the list when an opportinity raises.



(a) Start screen of *MediaServ* (b) Capturing a photo (c) Update content property (d) List of contents

Fig. 6: Application demonstration

We tested the app for three campaigns with a small number of users:

- Clean campus campaign: This campaign asked students to report dirts/littered objects spread around in our campus. This is a awareness campaign against reckless littering of objects in a campus.
- Festival campaign: This asked students to take photos of events of a week long student program known

as CSE FESTIVAL 2015. This asks for photos from within campus as well as within a certain period of time.

- Spelling campaign: Shops in cities frequently mispell common words in their displays. The purpose of this campaign is to collect pictures of these spelling mistakes in a nearby market area.

The followng table shows how many photos are taken per campaign and how many of them uploaded during the campaign duration (details in [18]).

Campaign	Generated	Uploaded
Clean campus campaign	31	21
Festival campaign	38	24
Spelling campaign	42	32

VI. Conclusion

In this paper, we mainly focus on service based image retrieval system. Evaluation technique of contents is illustrated. As users normally are not willing to give data in participatory network, hence a technique in where users can send content efficiently is illustrated in this text. Maximize the service campaigner's gain, and optimize the user's resource is mainly described. Efficient way of image processing in user device can be added in future, which will make the content evaluation process more efficient. We also tried to make an effective manner of using user bandwidth with two budgeting schemes. Various scheduling methods are described which can be used in an adaptive manner in different circumstances.

References

[1] "Emarketer." *http://www.emarketer.com/Article/Smartphone-Users-Worldwide-Will-Total-175-Billion-2014/1010536*.

[2] "Business insider." *http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9*.

[3] A. Rizvi, S. Ahmed, M. Bashir, and M. Y. S. Uddin, "Mediaserv: Resource optimization in subscription based media crowdsourcing," in *Networking Systems and Security (NSysS), 2015 International Conference on*, pp. 1–5, IEEE, 2015.

[4] Y. Jiang, X. Xu, P. Terlecky, T. F. Abdelzaher, A. Bar-Noy, and R. Govindan, "Mediascope: selective on-demand media retrieval from mobile devices," in *The 12th International Conference on Information Processing in Sensor Networks (co-located with CPS Week 2013), IPSN 2013, Philadelphia, PA, USA, April 8-11, 2013*, pp. 289–300, 2013.

[5] J. Redi, T. Hossfeld, P. Korshunov, F. Mazza, I. Povoa, and C. Keimel, "Crowdsourcing-based multimedia subjective evaluations: a case study on image recognizability and aesthetic appeal," in *ACM CrowdMM 2013*, (Barcelona, Spain), 10 2013.

[6] M. Y. S. Uddin, H. Wang, F. Saremi, G. Qi, T. F. Abdelzaher, and T. S. Huang, "Photonet: A similarity-aware picture delivery service for situation awareness," in *Proceedings of the 32nd IEEE Real-Time Systems Symposium, RTSS 2011, Vienna, Austria, November 29 - December 2, 2011*, pp. 317–326, 2011.

[7] T. H. Silva, P. O. S. V. de Melo, J. M. Almeida, J. F. S. Salles, and A. A. F. Loureiro, "A picture of instagram is worth more than a thousand words: Workload characterization and application," in *IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2013, Cambridge, MA, USA, May 20-23, 2013*, pp. 123–132, 2013.

[8] S. Hachem, V. Mallet, R. Ventura, A. Pathak, V. Issarny, P. Raverdy, and R. Bhatia, "Monitoring noise pollution using the urban civics middleware," in *First IEEE International Conference on Big Data Computing Service and Applications, BigDataService 2015, Redwood City, CA, USA, March 30 - April 2, 2015*, pp. 52–61, 2015.

[9] E. Kanjo, "Noisespy: A real-time mobile phone platform for urban noise monitoring and mapping," *MONET*, vol. 15, no. 4, pp. 562–574, 2010.

[10] N. Maisonneuve, M. Stevens, and B. Ochab, "Participatory noise pollution monitoring using mobile phones," *Information Polity*, vol. 15, no. 1-2, pp. 51–71, 2010.

[11] L. G. Jaimes, I. J. Vergara-Laurens, and M. A. Labrador, "A location-based incentive mechanism for participatory sensing systems with budget constraints," in *2012 IEEE International Conference on Pervasive Computing and Communications, Lugano, Switzerland, March 19-23, 2012*, pp. 103–108, 2012.

[12] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing," in *The 18th Annual International Conference on Mobile Computing and Networking, Mobicom'12, Istanbul, Turkey, August 22-26, 2012*, pp. 173–184, 2012.

[13] J. Lee and B. Hoh, "Sell your experiences: a market mechanism based incentive for participatory sensing," in *Eigth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom 2010, March 29 - April 2, 2010, Mannheim, Germany*, pp. 60–68, 2010.

[14] E. D. Cristofaro and C. Soriente, "Short paper: PEPSI - privacy-enhanced participatory sensing infrastructure," in *Proceedings of the Fourth ACM Conference on Wireless Network Security, WISEC 2011, Hamburg, Germany, June 14-17, 2011*, pp. 23–28, 2011.

[15] Q. Li, G. Cao, and T. F. L. Porta, "Efficient and privacy-aware data aggregation in mobile sensing," *IEEE Trans. Dependable Sec. Comput.*, vol. 11, no. 2, pp. 115–129, 2014.

[16] X. Yuan, X. Wang, C. Wang, A. C. Squicciarini, and K. Ren, "Enabling privacy-preserving image-centric social discovery," in *IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014, Madrid, Spain, June 30 - July 3, 2014*, pp. 198–207, 2014.

[17] Q. Li and G. Cao, "Providing efficient privacy-aware incentives for mobile sensing," in *IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014, Madrid, Spain, June 30 - July 3, 2014*, pp. 208–217, 2014.

[18] S. A. Minhajul Bashir, A.S.M Rizvi, "MediaServ: Resource Optimization in Subscription based Media Crowd-Sourcing." https://www.dropbox.com/s/jxuwpz9vn4hksb4/thesis-signed.pdf?dl=0/, 2015.