Proceedings of the 2019 Winter Simulation Conference
N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, eds.

A DIGITAL TV-BASED DISTRIBUTED IMAGE PROCESSING PLATFORM FOR NATURAL DISASTERS

Manuel Manriquez

Departamento de Ingenieria en Informatica
Universidad de Santiago de Chile
Santiago, 9170124, CHILE

Fernando Loor
Veronica Gil-Costa

CONICET, National Commission of Sc. and Tech.
Universidad Nacional de San Luis
San Luis, 5700, ARGENTINA

Mauricio Marin

CeBiB, Centre for Biotechnology and Bioengineering
DIINF, Universidad de Santiago de Chile
Santiago, 9170124, CHILE

ABSTRACT

After a natural disaster strikes people spontaneously respond by self-organizing, providing food and drink to the victims and to emergency response teams. During this process, people also share photos, messages and videos which can be used to improve the general understanding of the situation and to support decision-making. In this context, we propose to use digital television to create a community of digital volunteers who can help to identify objects inside images that cannot be processed automatically. Digital television can help to reach a larger number of digital volunteers because it can be easily used without installing special applications. We present a distributed platform composed of a server, a network of digital volunteers and an internet service provider. Our proposed platform aims to reduce the communication between the server and the digital volunteers and to reduce the workload of the server. We simulate our proposed platform with the peersim tool.

1 INTRODUCTION

Digital television (DTV) allows to broadcast high definition video and better sound quality than the analog signal. It also allows greater system versatility, multiple video signals, teletext, EPG (Electronic Guide of Programs), radio channels, panoramic image, interactive services, among others. Moreover, DTV includes features such as mobility and portability, which makes it available for different devices such as smartphones.

DTV is implemented with a Set Top Box (STB) connected to the television. The STB processes the compressed digital signal, decompresses it and sends it to the television. There are also televisions with STB integrated. STBs can be used: (1) to process low computation operations on user data, and (2) as a temporary storage for elements requested by users (Rosas et al. 2013).

Digital television allows users to interact with different services, for example, browse the electronic program guide or provide feedback to the broadcaster service. These services allow the user not only to be a receiver, but also to participate in the television program he/she is watching. This interaction occurs through the remote control associated with the STB, or through the smartphone if it receives the signal

One-Seg (Bordignon et al. 2009). Depending on the service, it is possible that this interaction generates events and/or requests over the Internet which must be processed by a server.

Upon a natural disaster such as earthquakes, tsunamis, volcano eruptions, hurricanes, tornados and floods, people share photos, messages and videos. In this context, DTV allows users to became digital volunteers. Users can collaborate by tagging images or voting an option from a list associated with the images that cannot be processed automatically. E.g. users vote to determine the category of the image such as infrastructure and service, affected people, emotional support, among others. The data processed by users can be collected and integrated into the management process to improve the general understanding of the situation or rescue actions. Some applications like the Digital Humanitarians (Verity and Meier 2012) and Tomnod (http://www.tomnod.com) allow digital volunteers to collaborate in the image processing process.

In this work, we propose to use digital television to create a community of digital volunteers who can help to identify objects inside images that cannot be processed automatically. We present a distributed platform composed by a server, digital volunteers (DTV users) forming a peer-to-peer (P2P) network and an Internet Service Provider (ISP). We aim to reduce the communication between the server and the digital volunteers and to reduce the workload of the server. The ISP routes the messages between the DTV users and the server. The server stores the images in a database usually kept in secondary memory, sends those images to the digital volunteers and selects the options associated with the images with the majority of votes. The digital volunteers process the images by selecting an option from the list of options of the images. Each digital volunteer has a local cache memory used to reduce the communication among the server and the volunteers. We present a dynamic scheme to improve the use of resources provided by the volunteers which includes a voting algorithm executed at the server side and a aggregation algorithm executed in background in the DTVs. Both algorithms rank the digital volunteers based on their processing speed and their communication latency. We simulate our proposed platform with the peersim tool (http://peersim.sourceforge.net/#pubs) which supports cycle-based and event-based simulations for P2P networks.

The remainder of this paper is organized as follows. Section 2 present related works. Section 3 details our proposed distributed platform. Section 4 presents the simulation setup and the results. Finally, Section 5 brings the conclusions and futures works.

2 RELATED WORKS

The greater the disaster the greater the response of people to cooperate (Twigg and Mosel 2017). This type of volunteer participation is known as crowdsourcing. Digital volunteers have helped to collect pertinent information much faster than officials or people in charge of coordination activities in natural disasters could do alone, with huge potential impacts on the responsibilities of officials in the management of the information. Emergency search and rescue teams already use pre-event remote sensing data when planning operations (Thorvaldsdottir et al. 2011).

The work presented in (Becker and Bendett 2015) describes some examples of how the Department of Defense of the United States uses crowdsourcing to give answers to problems of natural disasters. The authors concluded that there is a great benefit in taking advantage of the power of the crowd. The authors in (Barrington et al. 2012) presented a review of the state of the art on the use of crowdsourcing and analysis of images, particularly high resolution aerial. This work describes the experiences obtained in the cases of the earthquake in Haiti and in 2008 in China. The authors in (Ofli et al. 2016) proposed a hybrid scheme based on automatic techniques and crowdsourcing for aerial image processing. In this case, manual annotations are used to train a supervised learning system. However, this work does not describe the platform used and does not consider the interaction with information collected by people who are in the place of the event.

There are some platforms such as Tomnod (http://www.tomnod.com), GeoTag-X (Smith 2017), and some research works that address the problem of using volunteers for the processing of georeferenced images (Onorati and Díaz 2016;Diaz et al. 2016;Witjes et al. 2017;Turk 2017). In particular, Tomnod of the company DigitalGlobe is using Artificial Intelligence (AI) driven by crowdsourcing to automatically

identify the characteristics of interest in satellite and aerial images. Tomnod runs crowdsourcing campaigns, where volunteers support data mapping by validating the results of an image mining algorithm. GeoTag-X is a research project aimed at researching and evaluating collaborative on-line environments and software tools for creative learning.

Peer-to-Peer technologies have been used for digital TV services. The work presented in (Chang et al. 2010) used the information about the programs and time that the users watched those programs to make recommendations. BBC uses P2P to keep its programs broadcast for 7 days and to reduce storage on its servers (https://www.theguardian.com/media/organgrinder/2006/dec/21/bestofthebbctobereleased).

P2P cache schemes have been proposed mostly for Web pages (Rosas et al. 2012;Rosas et al. 2013). Approaches for cache mainly focus on optimizing one metric, such as hit rate, latency, or communication (Gummadi et al. 2003; Karagiannis et al. 2005; Hefeeda and Noorizadeh 2010).

In our work we focus on the latency, communication costs and on taking advantage of resources provided by the volunteers. Our proposal allows to process partial results inside the P2P network before sending the final results to the server. We rank the volunteer peers taking into account their processing speed and we select the slowest peers to process partial results in background.

3 A DIGITAL TV-BASED PLATFORM FOR IMAGE PROCESSING

In this section, we present our platform design for processing a large number of images. Our design focuses on distributing the computation among the available resources. Figure 1 shows the general scheme of our proposed platform. The server contains a database with images to process. For each image, we create a task with an image identifier (ID), the GPS coordinates and a list of options to vote. These tasks are inserted into a processing queue of the server. In the P2P network, each peer represents a volunteer (user) that will participate in the image processing process. Each peer has a local cache memory used to store images. The memory cache is used to reduce the communication between the P2P network and the server, to take advantage of the fact that the communication latency inside the P2P neighborhood is lower than the communication latency between the peers and the server. A percentage of the peers belonging to the P2P network are registered as volunteers. The remaining peers share their local cache but they do not participate in the image processing process.

Peers build an overlay network managed by the Internet service provider (ISP). In particular, our model follows a P2P Distributed Hash Table (DHT) (Rowstron and Druschel 2001). Internet service providers (ISPs) are responsible for delivering Internet access to clients from a given geographic area. To communicate with other ISPs, it is necessary to access the Internet backbone. The backbone is a shared network which enables communication among ISPs of the world. To make use of the network, ISPs must respect a Service Level Agreement (SLA) contract in which they commit to regulate their traffic to not compromise another ISPs communication.

3.1 Batch-Based Scheme for Image Processing

The DHT is used to partitioning the image space among the peers (Antonopoulos, Exarchakos, Li, and Liotta 2010). That is, the whole space of images is divided among the peers and each peer is responsible for a particular set of images. The steps involved in the image processing process are as follows:

1. The volunteer peers send a message to the server indicating they want to collaborate in the image tagging/voting process.
2. The server registers the peers as volunteers and sends a batch of N tasks to the peers. The same batch of tasks is sent to H peers. After a batch of tasks is sent to H peers, the server continues with the next batch of tasks.
3. The volunteers process each incoming batch of task as follows. First, they search for the image objects associated with the tasks inside the P2P network. To this end, the peers apply a hash function

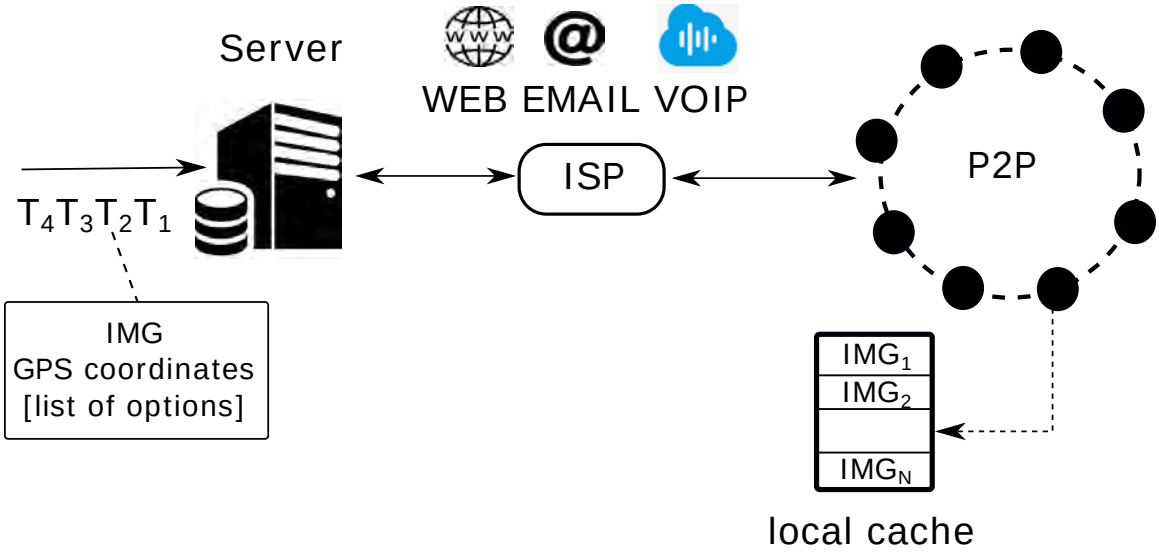

Figure 1: DTV-based distributed platform composed of a server, an internet service provider (ISP) and the P2P network. Each peer has a local cache to store images.

on the file name of the image objects to select the next hop inside the P2P network. Therefore, a message can be sent to several intermediate peers before arriving at the responsible peer.

4. The responsible peer searches for the image ID in its local cache memory. The cache memory implements an LRU replacement policy. If the image is found, then the responsible peer sends the image object to the volunteer peer. Otherwise, the peer responsible for storing the image in its local cache memory sends a message to the server.
5. The server searches for the images in the database (usually kept in secondary memory) and sends the image objects to the responsible peers.
6. The images are inserted into the local cache memory of the responsible peers and then sent to the volunteer peers.
7. The volunteer peers selects an option from a list associated with the image and send the results to an aggregation peer.
8. The aggregation peer collects the results from the volunteer peers and sends the final results to the server.
9. The server ranks the votes of the options associated with the tasks and evaluates whether an option has the majority of votes. E.g., if an option receives more than 60% of the votes. If so, the task is finished and the results are stored in the database. Otherwise, the tasks are re-sent to H peers.

These steps are illustrated in Figure 2 where each peer is responsible for a different set of images (square, circle and the different types of stars). In this example, the $peer_1$ requests tasks to the server (step 1). The server sends a batch of tasks to $peer_1$ (step 2). $Peer_1$ searches for the image associated to the first task. In this example it searches for a blue square. Then it sends a message to $peer_2$ which is the next hop selected by the DHT (step 3). $Peer_2$ does not have the requested image, so it sends the message to the next hop in the P2P network. In this case, the message reaches to $peer_3$ which is the responsible peer for this kind of images. $Peer_3$ does not have the blue square image in its local cache. Therefore, it sends a message to the server (step 5). The server retrieves the images form the database and sends the requested image to the $peer_3$. $Peer_3$ inserts this new image into its local cache memory and sends the image to the volunteer $peer_1$ (step 6). $Peer_1$ votes for an option from a list associated with the incoming image and sends the result to $peer_4$ which collects and merges the H results for a given task (step 7). Finally, $peer_4$

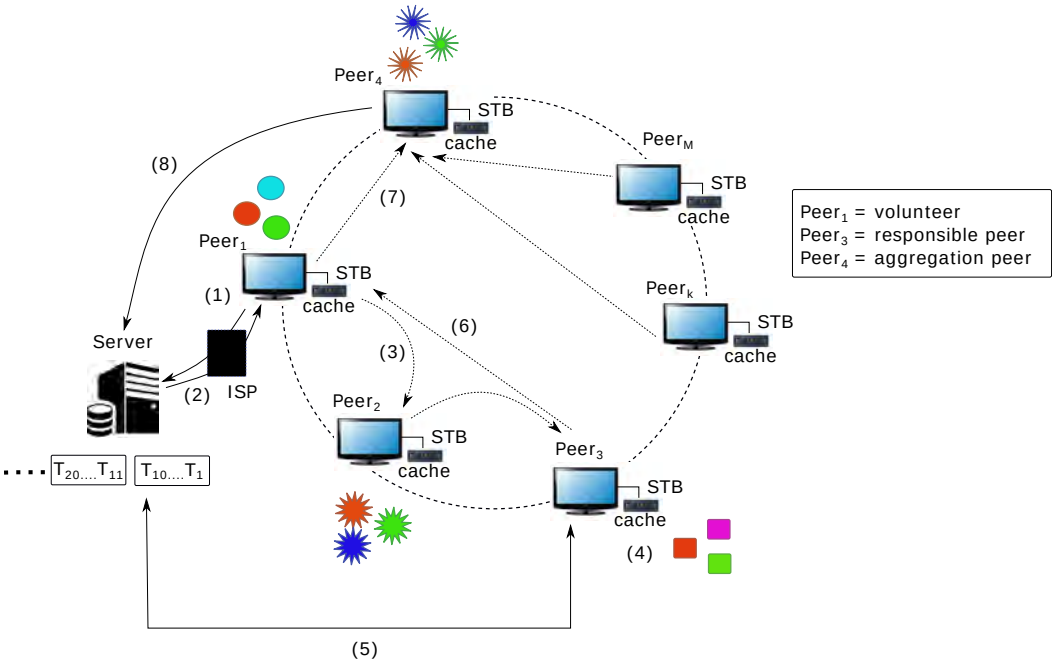

Figure 2: Sequence of steps executed in the distributed platform. Each peer can play different roles: volunteer, responsible for a set of images and aggregation peer.

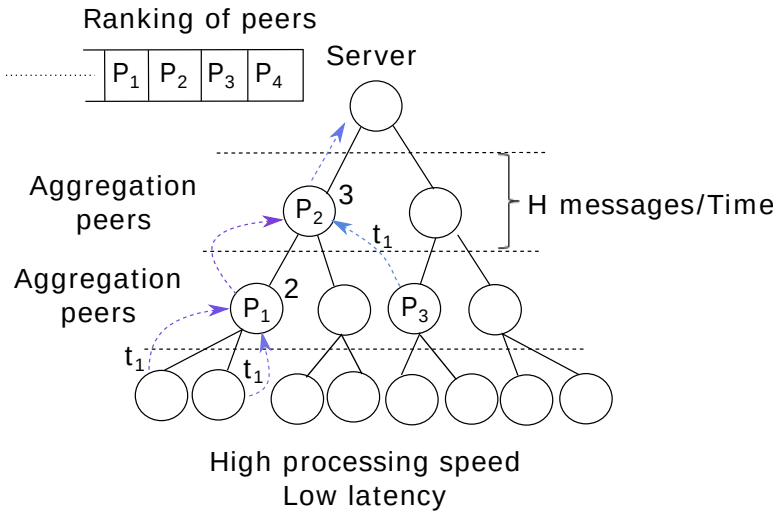

Figure 3: Aggregation scheme of partial results inside the P2P network.

sends the results to the server. Although it is not detailed in the sequence of steps described above, the communication between the peers and the server is made through the ISP.

The aim of the aggregation peers is to reduce the communication traffic between the P2P network and the server. The communication inside the P2P network has a lower latency than the communication between the peers and the server. Volunteers with a high processing speed and lowest communication latencies are marked as *efficient* peers and peers with low processing speed are used to compute the majority of votes in background. To this end, we use a binary tree as detailed in Figure 3. Each node represents a volunteer peer. The leaves are the efficient peers and the root is the server. In Figure 3, the $peer_1$ receives 2 messages

of the same task t_1, merges the votes and sends a message to the $peer_2$. Then, the $peer_2$ also receives a message from $peer_3$ with results of the same task t_1. The $Peer_2$ merges the votes for task t_1 (3 results in total) and sends a message to the server. The tree is updated every Δ_t units of time. To this end, the server collects statistics about the processing speed and the communication traffic among the peers. A volunteer peer sends its results to its parent node in the tree. When a message with results arrives to an aggregation peer it creates a temporary data structure to store partial results of the same tasks. When the aggregation peer receives H result messages for a given task or a time T has elapsed, it merges the partial results and sends a message to its parent node in the tree.

Algorithm 1 Voting algorithm executed in the server.

```
1:  task = input_queue.select( task_id )
2:  finish = false
3:  if task.number_of_results > 1 then
4:     Merge(task.results)
5:  end if
6:  for  i = 0 TO i < msg.option.size()  do
7:     if task.option[i].votes/H >= threshold  then
8:        task.option[i].select = true
9:        finish = true
10:    end if
11: end for
12: if finish OR task.iteration >= Limit then
13:    DB.store(task.img_id, task.results)
14:    exit
15: end if
16: task.iteration++
17: for i = 0 TO number_of_volunteers do
18:    Sc[i] = (1/s2p + 1/p2s) + AvgTasks() + TasksSolved/Tasks + Correct/Tasks
19: end for
20: Sort(Rc)
21: for  i = 0 to H  do
22:    p = Sc[i]
23:    Send(p,task)
24: end for
```

3.2 Voting Algorithm

For each image, the server receives messages with the votes of the users and executes a voting algorithm. The server receives a single message if all H votes were processed within an aggregation peer. Otherwise, it can receive several messages with results for the same task.

Algorithm 1 shows the main operations executed by the voting algorithm. First, the server retrieves all results for a given *task* from the *input_queue*. The merge operation, in line 4, computes the number of votes for each option of the task. From line 6 to 11, if an option has the majority of votes the algorithm marks the option and indicates that the task is finished. The total number of votes for an option has to be higher than a threshold (line 6), e.g. the number of votes for an option has to be higher than 60% of the total number of votes. When the total number of votes for a given option is higher than the threshold, we say the task has consensus. Then, in lines 12-15, the algorithm stores the results in a database. We put a

Limit on the number of times a task can be re-sent to the peers when the threshold is not reached by the majority of votes. If the task is not finished and the $task.iteration < Limit$, we select H new volunteers. To this end, for each peer i the algorithm computes a score in $Sc[i]$ taking into account the communication latency between the server and the peer i ($s2u$ and $p2s$), the average processing time of the peer $AvgTasks()$, the average number of tasks processed by the peer $TasksSolved/Tasks$ and the average number of correct votes of the peer $Correct/Tasks$. Finally, the task is re-sent to the H peers with highest scores.

4 EXPERIMENTS

We evaluated the performance of our distributed platform composed of a server, the network of volunteer forming a P2P network and the ISP. We built a simulator with the peersim tool that implements a transport layer and a P2P overlay. Chord (Stoica, Morris, Liben-Nowell, Karger, Kaashoek, Dabek, and Balakrishnan 2003) has been used as the overlay network in our experimentation. Each STB has a memory of 500 MB and we use only 10% as cache memory. The remaining 450MB are used to deploy the operated system and other applications.

Experiments were performed for a total of 13.233 images of faces obtained from the labelme repository (https://en.wikipedia.org/wiki/LabelMe). The service times were set according to the statistics presented in (Meier 2013). In particular, the service time of the digital volunteers is a value between $[100ms, 20sec]$. Each task has a list of four options.

4.1 Performance Evaluation

In the following experiments we evaluate the communication latency and the computation time reported by our proposed distributed platform. To this end, we simulate a network of 1000, 2500 and 5000 peers. However, only a percentage of the peers register as volunteers. The remaining peers share their local cache memory, but they do not participate in the image voting process. At the beginning of the simulation, all the batches of tasks are sent to the P2P network. Each task is sent to $H = 10$ digital volunteers. If the task has to be re-sent to the P2P network because there is no consensus about the options associated to its image, we set $H = 5$ and $Limit = 4$. We set $T = 20$ seconds in the aggregation peer. That is the time elapsed before sending the results to the parent node in the tree. We compare the performance achieved with our proposal against a baseline distributed platform. The baseline does not include cache memories allocated in the peers neither aggregation peers, and the voting algorithm selects the next H peers at random.

In Figure 4 the x-axis shows the simulation time advance in milliseconds. The y-axis shows the computation (service) time reported by the server for different P2P network configurations. Figure 4(a) shows results for the baseline and Figure 4(b) shows results for the proposal with different network sizes $[1000, 2500, 5000]$, the voting threshold$= 60\%$ and the percentage of digital volunteers is 50% of the total number of peers (e.g. with a network size of 1000 there are 500 digital volunteers). In Figure 4(c) and Figure 4(d) we set the network size to 2500 peers, the threshold$= 60\%$ and we experiment with different percentage of digital volunteers $[50\%, 60\%, 70\%]$. In Figure 4(e) and Figure 4(f) we set the network size to 2500 peers, the percentage of digital volunteers is 50% of the P2P network and we experiment with different threshold values $[60\%, 70\%80\%]$.

All figures present peaks of service times which represent the secondary memory accesses to recover the images from the database. The baseline presents peaks of service times close to 320ms meanwhile our proposal shows lower peaks close to 240ms, because the cache memory used in our proposal reduces the number of secondary memory access by 20% in average. Additionally, after accessing the database the baseline reports service time close to 30ms. and our proposal reports service times of 5ms. in average. This is because the aggregation peers reduce the computation operations executed in server.

In Figure 4(a) and Figure 4(b) there is a larger delay between the peaks of service times as we increase the network size, because all batches of tasks are sequentially sent to the digital volunteers who request the image objects for each task. In the baseline there is no cache memory, therefore all the images are

Table 1: Total number of messages and bytes transmitted for each type of message between the server and the peers.

	RQST BoT	SEND BoT	RQST IMG	SEND IMG	RESULTS	MORE
Number of messages						
Proposal	1250	1250	17646	17646	1378	885
Baseline	1250	1250	30530	30530	1654	2010
Bytes						
Proposal	58464	151641	952884	250286907	151762	51890
Baseline	58464	151641	1648620	431893335	177432	156540

sequentially requested to the server who accesses to the database. Figure 4(c) and Figure 4(d) show the same behavior as we increase the percentage of peers registered as digital volunteer. Finally, in Figure 4(e) and Figure 4(f), as we increase the threshold values the total image processing time (showed in the *x*-axis) also tends to increase as each task is re-sent many times before reaching a majority of votes or the *Limit* of the iterations.

Figure 4 shows that our proposal can reduce the computation time in the server, however the time to process all images tends to increase by 7% in average. This is because, in our proposal, volunteers search for images inside the P2P network by sending messages that perform several hops before reaching the responsible peer.

Figure 5 shows the communication between the server and the volunteer peers as the simulation time advance with different parameter configurations. Figure 5(a) shows results obtained with our proposal and Figure 5(b) shows the results obtained with the baseline strategy. The baseline shows that the communication tends to persist high as the simulation time advance with some peaks reaching more than 600000 bytes per unit of time.

Figure 6 shows the total communication cost measured as the total number of bytes transmitted between the server and the peers with different parameter configurations. Each bar represents a different parameter configuring $[network\_size, volunteers(\%), threshold(\%)]$. Results show that the P2P network size drastically impacts on the communication costs. The larger the network the greater the number of bytes transmitted. In the best case, our proposal reduces by 42% the communication costs as we increase the network size. In general, our proposal reduces the communication costs between the server and the peers by 39% in average.

Table 1 shows the number of messages and the amount of bytes transmitted between the server and the peers for different types of messages:

- RQST BoT: The peers request a batch of tasks to the server.
- SEND BoT: The server sends a batch of tasks to the volunteer peer.
- RQST IMG: The responsible peer requests the image object to the server.
- SEND IMG: The server sends the image object to the responsible peers.
- RESULTS: The volunteer peers send the votes for each task to the server.
- MORE: The server re-sends the tasks without consensus.

Results show that our proposal reduces by 42% the number of messages requesting/sending image objects. Therefore, the amount of communication bytes is also drastically reduced. Our proposal processes partial results in the slowest peers in background, thus the number of messages with results sent to the sever is less than in the baseline strategy. This is also reflected in the amount of communication bytes. Finally, the last column of Table 1 shows the number of messages involved in additional iterations when the majority of votes is not obtained. Our proposed voting algorithm reduces by 55% the number of messages and by 66% the communication bytes.

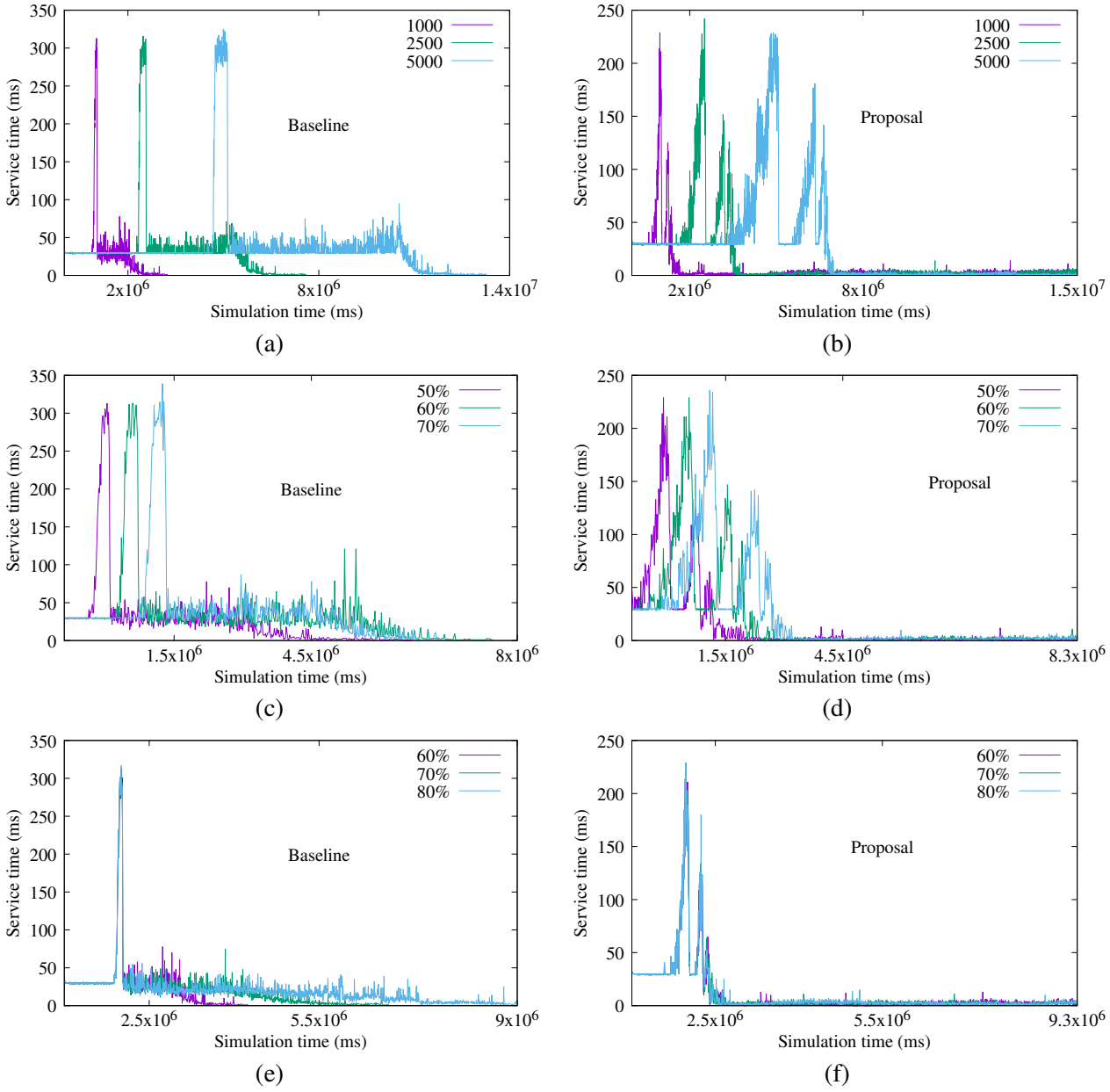

Figure 4: Service time reported by the server with different parameter configurations: (a) and (b) P2P network size [1000,2500,5000]. (c) and (d) digital volunteers of the network [50%,60%,70%]. (e) and (f) threshold values [60%,70%,80%].

Finally, in Figure 7 we show the communication cost inside the P2P network. That is, the communication between the peers and the communication between the peers and the ISP. A log scale is used for the *y*-axis. The *x*-axis shows the simulation time in milliseconds. As expected, our proposal (Figure 7(a)) reports higher number of bytes transmitted inside the P2P network than the baseline strategy (Figure 7(b)). That is because our proposed strategy searches the image objects in the local cache memories of the peers.

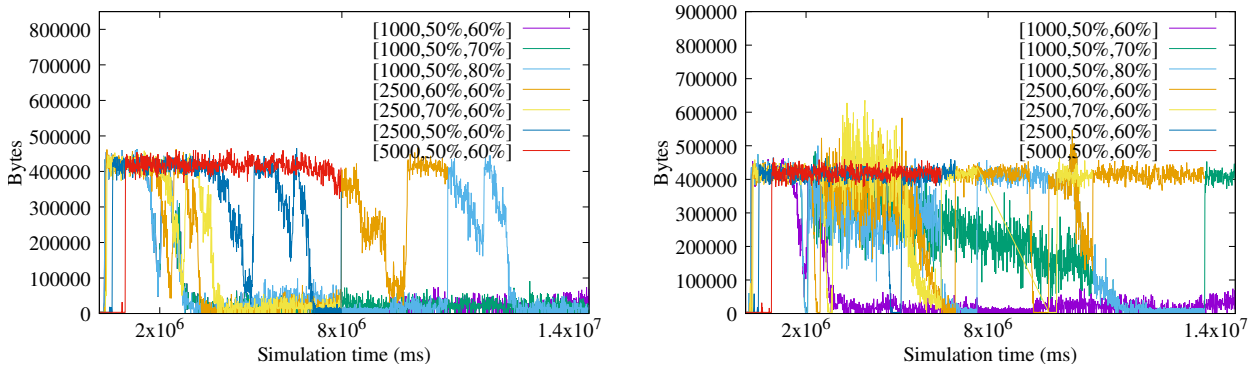

Figure 5: Communication cost reported by our proposal and the baseline with different parameter configurations as simulation time advance: (a) Proposal and (b) Baseline.

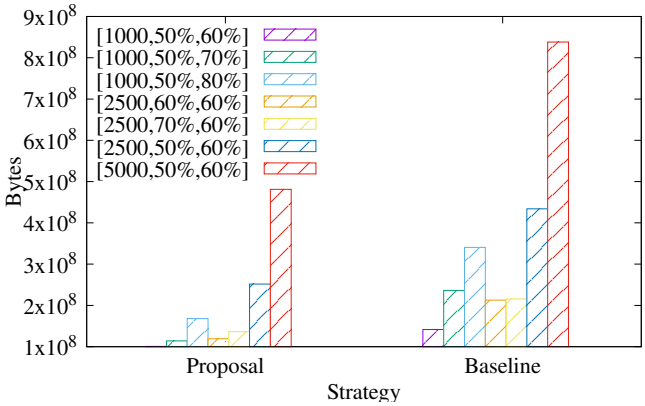

Figure 6: Total communication cost reported by our proposal and the baseline with different parameter configurations.

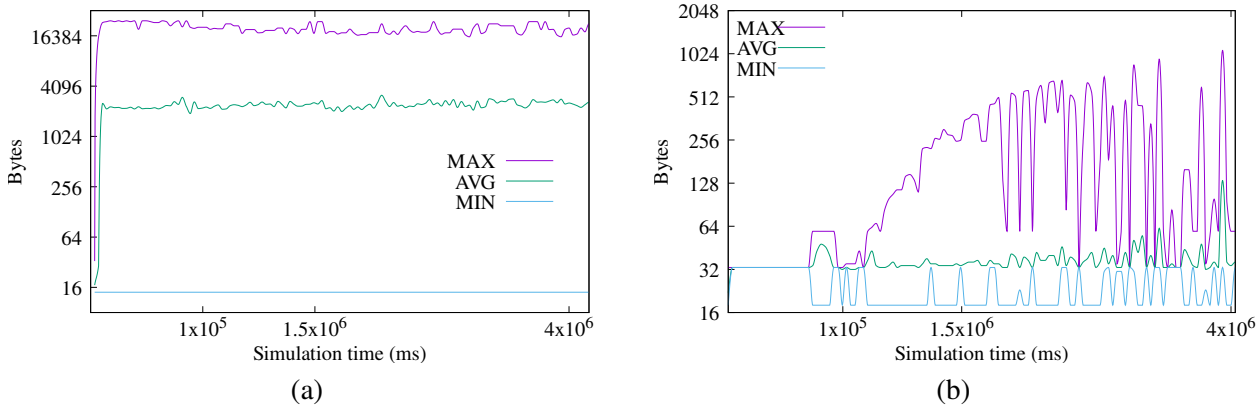

Figure 7: Communication cost inside the P2P network: (a) Proposal and (b) Baseline.

5 CONCLUSIONS

In this work, we proposed a distributed platform to process images shared by users when a natural disaster strikes. The platform includes a network of digital volunteers connected to a server through an ISP. Digital volunteers collaborate by accessing to Digital TV services thought STB and form a community with low communication latencies. Our proposal includes two algorithms to rank the digital volunteers taking into account their processing speed and their communication latencies. The first algorithm runs at the server side and selects the new digital volunteer to process an image when there is no consensus about the options associated with the images. The second algorithm executes in background the processing of partial results to take advantage of the resources provided by the P2P network.

We developed a simulation framework using the peersim tool. We evaluated the performance of our proposal with different parameter configurations. Results show that our proposal can reduce the computation costs at the server side and the communication between the server and the community of digital volunteers at the cost of incurring into a small delay to finish the tasks.

As future work, we plan to evaluate the performance of our proposal with a dynamic P2P network. We also plan to evaluate the impact of each parameter in the performance of our platform and to auto adjust the most relevant parameters taking into account different communication traffic conditions.

ACKNOWLEDGMENTS

This research was supported by the supercomputing infrastructure of the NLHPC Chile and partially funded by CONICYT Basal Funds FB0001, Fondef ID15I10560.

REFERENCES

Antonopoulos, N., G. Exarchakos, M. Li, and A. Liotta. 2010. *Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies, and Applications*. Information Science Reference - Imprint of: IGI Publishing.

Barrington, L., S. Ghosh, M. Greene, S. Har-Noy, J. Berger, S. Gill, A. Lin, and C. Huyck. 2012. "Crowdsourcing Earthquake Damage Assessment Using Remote Sensing Imagery". *Annals of Geophysics* 54(6):680–688.

Becker, D., and S. Bendett. 2015. "Crowdsourcing Solutions for Disaster Response: Examples and Lessons for the US Government". *Procedia Engineering* 107(1):27 – 33.

Bordignon, A., S. A., F. Varella, J. Toss, V. Roesler, and M. Barbosa. 2009. "Mechanisms for Interoperable Content Production Among Web, Digital TV and Mobiles". *INFORMATICA NA EDUCACAO: teoria & pratica* 12(1):329–350.

Chang, J.-H., C.-F. Lai, Y.-M. Huang, and H.-C. Chao. 2010. "3PRS: a Personalized Popular Program Recommendation System for Digital TV for P2P Social Networks". *Multimedia Tools and Applications* 47(1):31–48.

Diaz, P., J. M. Carroll, and I. Aedo. 2016. "Coproduction as an Approach to Technology-Mediated Citizen Participation in Emergency Management". *Future Internet* 8(41):1–16.

Gummadi, K. P., R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. 2003. "Measurement, Modeling, and Analysis of a Peer-to-peer File-sharing Workload". *ACM Special Interest Group on Operating Systems Operating Systems Review* 37(5):314–329.

Hefeeda, M., and B. Noorizadeh. 2010. "On the Benefits of Cooperative Proxy Caching for Peer-to-Peer Traffic". *IEEE Transactions on Parallel and Distributed Systems* 21(7):998–1010.

Karagiannis, T., P. Rodriguez, and K. Papagiannaki. 2005. "Should Internet Service Providers Fear Peer-assisted Content Distribution?". In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, 6–6. Berkeley, CA, USA: USENIX Association.

Meier, P. 2013. "MicroMappers: Microtasking for Disaster Response". https://irevolutions.org/2013/09/18/micromappers/, accessed 15th August 2019.

Ofli, F., P. Meier, M. Imran, C. Castillo, D. Tuia, N. Rey, J. Briant, P. Millet, F. Reinhard, M. Parkan, and S. Joost. 2016. "Combining Human Computing and Machine Learning to Make Sense of Big (Aerial) Data for Disaster Response". *Big Data* 4(1):47–59.

Onorati, T., and P. Díaz. 2016. "Giving Meaning to Tweets in Emergency Situations: a Semantic Approach for Filtering and Visualizing Social Data". *SpringerPlus* 5(1):1782–1782.

Rosas, E., N. Hidalgo, and M. Marín. 2012. "Two-Level Result Caching for Web Search Queries on Structured P2P Networks". In *Proceedings 18th IEEE International Conference on Parallel and Distributed Systems, Singapore, December 17-19, 2012*, 221–228.

Rosas, E., N. Hidalgo, M. Marin, and V. Gil-Costa. 2013. "Web Search Results Caching Service for Structured P2P Networks". *Future Generation Computer Systems* 30(1):254–264.

Rowstron, A., and P. Druschel. 2001. "Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems". In *Middleware 2001*, edited by R. Guerraoui, 329–350. Berlin, Heidelberg: Springer Berlin Heidelberg.

Smith, C. 2017. *A Case Study of Crowdsourcing Imagery Coding in Natural Disasters*, 217–230. Springer International Publishing, edited by S. Hai-Jew.

Stoica, I., R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. 2003. "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications". *IEEE/ACM Transactions on Networking* 11(1):17–32.

Thorvaldsdottir, S., E. Birgisson, and R. Sigbjornsson. 2011. "Interactive On-Site and Remote Damage Assessment for Urban Search and Rescue". *Earthquake Spectra* 27(S1):S239–S250.

Turk, C. 2017. "Cartographica incognita: Dijital Jedis, Satellite Salvation and the Mysteries of the Missing Maps". *The Cartographic Journal* 54(1):14–23.

Twigg, J., and I. Mosel. 2017. "Emergent Groups and Spontaneous Volunteers in Urban Disaster Response". *Environment and Urbanization* 29(2):443–458.

Andrej Verity and Patrick Meier 2012. "Digital Humanitarians". http://digitalhumanitarians.com/, accessed 15th August 2019.

Witjes, N., P. Olbrich, and I. Rebasso. 2017. *Big Data from Outer Space: Opportunities and Challenges for Crisis Response*, 215–225. Springer Vienna, edited by C. Al-Ekabi, B. Baranes, P. Hulsroj,and A. Lahcen.

AUTHOR BIOGRAPHIES

MANUEL MANRIQUEZ L. is a master student at Universidad de Santiago de Chile (USACH), currently working at Centro de Innovacion en Tecnologias de la Informacion para Aplicaciones Sociales (CITIAPS). His email address is manuel.manriquez@usach.cl.

FERNANDO LOOR received his degree in Electronic Engineering (2016) at Universidad Nacional de San Luis (UNSL), Argentina. He is a PhD. student in Computer Science at UNSL. He holds a scholarship from CONICET. He is also a professor assistant in the courses of "Signals and Systems" and "Digital Signal Processing" at the UNSL. His email address is floor@unsl.edu.ar.

VERONICA GIL-COSTA received her MSc (2006) and PhD (2009) in Computer Science, both from Universidad Nacional de San Luis (UNSL), Argentina. She is a former researcher at Yahoo! Labs Santiago hosted by the University of Chile. She is currently an associate professor at the University of San Luis and researcher at the National Research Council (CONICET) of Argentina. Her email address is gvcosta@unsl.edu.ar.

MAURICIO MARIN is a former researcher at Yahoo! Labs Santiago hosted by the University of Chile, and currently a full professor at University of Santiago, Chile. He holds a PhD in Computer Science from University of Oxford, UK, and a MSc from University of Chile. His research work is on parallel computing and distributed systems with applications in query processing and capacity planning for Web search engines. His email address is mauricio.marin@usach.cl.