# How Should My Device Behave Now? Adapting Consensus Protocols for Autonomous Context Management

Fátima Castro-Jul, Ana Fernández-Vilas, Rebeca P. Díaz-Redondo
Universidade de Vigo, Campus Universitario s/n, 36310 Vigo, Spain.

* Corresponding author. Email: {fatima, avilas, rebeca}@det.uvigo.es
Manuscript submitted October 5, 2014; accepted March 18, 2016.
doi: 10.17706/jcp.12.3.200-211

**Abstract:** Managing context-dependent behavior in devices that are constantly on the move is a laborious task, especially when it comes to situations users (and therefore devices) are unfamiliar with. Due to the impossibility of using predefined rules or previous training, static and machine-learning context-awareness methods are impractical. Although collaboration with nearby peers represents a promising alternative, research on this area has been scarce. In this paper we propose a distributed consensus protocol, specifically designed to enable device cooperation with adaptation purposes. The protocol has been designed and implemented and its performance has been demonstrated through simulation.

**Key words:** Context-Awareness, device adaptation, distributed communications, mobile devices, proximity-based communications.

## 1. Introduction

Users carry their smartphones everywhere and, therefore, they require them to adapt to multiples contexts. This demand for device adaptation will increase as more highly-mobile devices, such as wearables, are becoming popular. There exist many proposals on inference of everyday places and studies on how users behave or want their devices to behave [1]-[3]. However, devices' behavior in situations different from usual routine remains an open issue. Asking others when in doubt is a behavior pattern employed by humans that could also benefit their devices [4], [5]. It can be used as a complement for other context-based applications or schedulers, either static or machine learning. When these systems find themselves in an unknown situation, they apply a default behavior or ask the user directly [6]. Employing a voting system instead improves their autonomy and their adaptation success. On the other hand, most context adaptation systems are based on context modeling [7] and, therefore, require modeling languages or similar tools. However, what devices actually need is information on how to perform certain actions rather than on general context. Context information is used ultimately to decide to enable/disable features on the smartphone according to modes. That is to say, context is a tool to ease feature management. Generally, a mode for every context is created. That is useful for usual contexts but not for non every-day situations, as the same adaptation is not required often. As a result, instead of creating a mode for each single one, they are normally sorted into classes such as ''entertainment'', ''gym'' or ''shop''. However, this may be too general. A better approach could be to decide directly about the features, skipping the step of detecting and modeling a context that may not be accurate enough or that it is not going to be used again.

On the whole, taking behavior decisions following advice on specific features from nearby peer devices is

a context managing strategy that can be employed in unusual situations. Therefore, it is necessary to develop coordination mechanisms that enable a quick and efficient collaboration between peers. In order to meet this need we propose a proximity-based distributed protocol, whose description and simulation we present in this paper. As a result of their proved value as a collaboration tool, we consider that consensus protocols are an interesting mechanism to be employed in context awareness and therefore, our model is based on them. In the following section we start by describing our scenario and assumptions. Later in Section 3 we detail our proposal. It is followed by several suggestions for optimization (Section 4) and implementation details (Section 5). In Section 6, we supply the results of the protocol simulation, using data gathered in real situations. Later, in Section 7, we discuss simulation results and we compare our protocol with related work. Finally, we present the conclusions of our work, together with the outline of some topics for future work (Section 8).

## 2. Scenario

From physical distance to noise level, current sensors in smartphones can measure almost everything in their surroundings. GPS sensors, accelerometers, magnetometers, gyroscopes, light sensors and microphones can be found nowadays in almost every smartphone or tablet. They can be accompanied by thermometers, hygrometers, heart rate or even radiation level sensors. Not forgetting those related to communication technologies such as Bluetooth and WiFi, which can be employed to measure distances and detect nearby devices.

In this paper, we consider device context, defined as the input and inference about physical environment lead by devices' sensors. Different groups of them have been utilized to detect location and location changes [1], [8], [9] Here, we do not consider which sensors to use or how to discern if the surroundings have changed, we just focus on how to communicate with nearby devices when this change has been detected. In other words, collaboration is triggered by a detection service that uses one or several of the sensors listed above and that is not examined in this paper. Collaboration could be triggered at any context change or at changes to situations devices have no previous knowledge about. For simulation purposes we will take the former approach so that every node will request the others' help every time a change is detected.

Considering devices' communication mechanisms that allow them to easily interact with their peers, we find mobile devices capable of performing context-related information exchange and adaptation tasks by using only their existing features. As a result, a context-aware system should not include additional requirements so as to be easily implementable and usable. That applies both to traditional systems and novel approaches focused on collaboration. For the latter it is especially interesting in order to make the collaboration scheme compatible with multiple devices. The procedure should include the following steps, based on consensus schemes:

- New device requests others' opinion.
- Neighbor devices reply with their current state.
- Requester device sends its decision so other nodes can employ it for possible decision changes.

Neighborhood can be defined in terms of physical distance, social closeness or any kind of relationship considered useful for the collaboration. For instance, neighbor devices could be those who have visited similar places or whose users share interests. As a first approach, in this paper we consider physical neighborhood. A neighbor node is any node located in the requester's communication range.

By using physical proximity, we avoid possible privacy issues that may discourage users from employing the system. Physical distance has been found to be one of factors considered by users to decide whether to disclose their data [10], [11]. This applies specially to location-dependent information due to the fact

nearby devices are likely to have access to it themselves. Hence, since behavior is location-dependent it is not a significant privacy issue to let devices in the same room know which behavior users consider appropriate.

Considering proximity, a transmission technology able to reach every device in a room-size area in one hop will serve the purpose while allowing us to keep the protocol simple. Simplicity is a main feature as changing circumstances require fast adaptation. Also because of that, the protocol should be distributed so as to allow direct communication between peers. Moreover, a distributed system prevents privacy risks that may arise when a third party is involved.

Our system is not designed for a determined situation but to be flexible enough to be used in any. We assume high mobility (devices are going in and out of the area at random times) and the existence of a *right* decision nodes will converge to. This decision will be the one most nodes have chosen and will change as soon as nodes change it due to context evolution. We do not consider other requirements for device behavior so it can mostly be based on users' preferences and agreements. Collaboration should not be regarded as an intrusive mechanism that will be imposed to users. At any time, any user can modify the behavior decided by cooperation to apply a different one of their choice. This user's alternative will be then disclosed to the others and may eventually lead to a change in the consensus behavior.

Even though this is not a classical consensus scenario due to the fact the number of participants is unknown and constantly varying, we employ similar procedures for opinion collection and decision propagation.

## 3. Protocol Description

The consensus protocol comprises the already mentioned three steps: discovery and request, reply and decision propagation. The first two are employed to make decisions while the last relates to decision reconsideration when circumstances change. Since we plan to use a transmission technology able to reach every device in a room in one-hop, the protocol is quite simple.

### 3.1. Decision Making

Discovery of nearby devices and information request are performed in a single step. Every node broadcasts its request. Requests could be sent unicast to known neighbors but, in that way, nearby neighbors should be sent periodical messages in order to get nodes always informed on their current neighborhood. Limiting the communication to request times, the number of messages in the network decreases and network congestion is avoided. We work on the assumption that requesters will not choose to whom they will consult and neighbors will collaborate by default. We consider neighbors the nodes that are close enough to answer our broadcast requests.

Format for request frames is shown in Fig. 1(a) and includes the fields: message type and sender IP.

Once other nodes have received a request they answer immediately. Basic answer frame is shown in Fig. 1(b). Every node votes according to its current state, which fills the reply info field.

In other consensus protocols, nodes make a decision once they have received votes from a certain number of neighbors [12]. As we work with highly-mobile nodes we set a time limit instead. The requester node will wait a certain time $Y$ and then it will set its decision as the one that repeats the most in the received replies. In case of draw, decision will be randomly chosen between the most voted ones.

Replies received after $Y$ will be discarded. $Y$ must be fixed and will be assessed through simulation.

### 3.2. Decision Reconsideration

Once a decision is made, it is broadcast through the whole network in order to be used for learning. Thus, node isolation and its subsequent impossibility to know when to change its behavior are prevented. Packet

structure is shown in Fig. 1(c). Another option may be to send unicast decisions only to nodes that have sent replies, instead of broadcasting them, so as to diminish the network load. Nodes can use received decision to reconsider their own state. This learning is necessary because devices' states should change if the context does. As a result, nodes pay attention to all decision they receive, compare them with their own status and with other received decisions. If the received decision matches current state it is dropped. To decide when to change their status nodes use *the Majority of freshest X with threshold* [13]. This strategy requires the reception of a certain number of votes X in a certain time threshold. These restrictions guarantee that changes in decisions are always performed according to the current situation. Nodes are required to save all received decisions during the threshold time to compare with other received decisions. Time threshold should be fixed and defined beforehand and it is the reason that makes necessary to send timestamps in decision packages. The number of necessary votes X should be dynamically changed according to the number of neighbors as suggested in [13]. However, being aware of the current number of neighbors for X calculation would require an independent neighbor discovery mechanism. As we do not implement such as system, we will use a fixed X, which will be assessed through simulation.

This learning is just short-term, it is only used as long as the device remains in the place. Once the device leaves the area it discards all information related to it. In order to remember it for future behavior patterns an extra system should be designed for long-term learning.

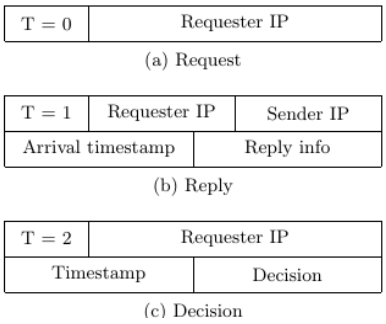



Fig. 1. Message frames used in the protocol.

| | | | |
|---|---|---|---|
| Calls | Messaging | Silence | Mobile use |
| Camera | Social | Location | Uploads |

Fig. 2. Standard feature vector for devices' state communication.

## 4. Optimization

In order to improve the protocol performance and reduce the amount of message that are sent, several optimization measures are proposed in this section. As in the case with parameters mentioned in the previous section, we provide no specific values for the variables described. They will all be assessed through simulation.

- **Request Replacement with Learning.** When a new node arrives, it will start receiving decisions from previous voting processes. If the number of those is large enough, the node may learn directly from nodes' decisions them instead of sending a request. In that way, less messages are needed without compromising the protocol performance since those contain the same information as the replies they would send if they were requested. In case the number of received decisions during a certain time is small, the node sends a request.
- **Informing Others on Decision Changes.** Nearby nodes are likely to have received the same decisions and therefore, to change decision at the same time. However, state contagion may be faster if every node broadcast its decision when it is changed. That would be especially useful in those cases where a node's state is not changed because of received decisions but directly by the user. These decisions will always be broadcast as recent list of nearby nodes is not known.

- **Dynamic X.** Due to the lack of a neighbor detection mechanism, we have used a fixed X in the basic scheme simulations. However, there exist a method to infer network density and therefore, dynamically calculate X. It consists on sending the number of received replies in decision messages. Thus, every receiver has a recent neighbor count and they update their status taking X as half the neighbors plus one. If decisions packets are sent to inform on decisions changes made on user's request, this field is left blank and X is not updated. Even if this strategy did not improve network load it would allow protocol adaptation to different circumstances, eliminating the need for X assessment.
- **Random Replies.** In case there is a high density of nodes in the room and assuming most of them share the same state it may not be necessary that every nodes replies. Nodes randomly decide whether to reply or not according to a certain probability in order to decrease network load.

## 5. Implementation

Previous protocol description lacks details necessary for practical implementation. Changes do not compromise the protocol's performance and should be performed in order to adapt the protocol to different devices or target scenarios.

So as to keep the protocol simple, a transmission technology whose range includes a room-size area is required. Moreover, we limit our choice to technologies currently available in most commercial devices. As a result, there were only two options available: Bluetooth and WiFi. WiFi was chosen over Bluetooth due to its usability to measure distances between devices, which we plan to use in the future to implement a weighted voting scheme.

Reply information sent by nodes on reply messages consists on a specific set of features for general requests, which needs to be defined to provide a standard format known by all the nodes. An example of feature vector can be found in Fig. 2. It is a 8-bits field where every bit represents the vote/state of a node and it represents the smartphone state. Every bit can take two values that express yes/no to each feature. The vector considers only very general features that apply to most smartphones nowadays. These features can be adapted to different devices or new smartphone services. Most of these features include modification/blockage of smartphone features, which are automatically performed by the device itself when the decision is applied. Users will receive a notification when they try to perform one of the blocked actions. Behavior applicable to other applications could be inferred from these. For instance: playing games/entertainment can be included in mobile phone usage.

## 6. Simulation

In order to analyze the performance of the protocol, real data was gathered from two different scenarios, one of them with high-mobility (university cafeteria) and another one with low-mobility (conference sessions).

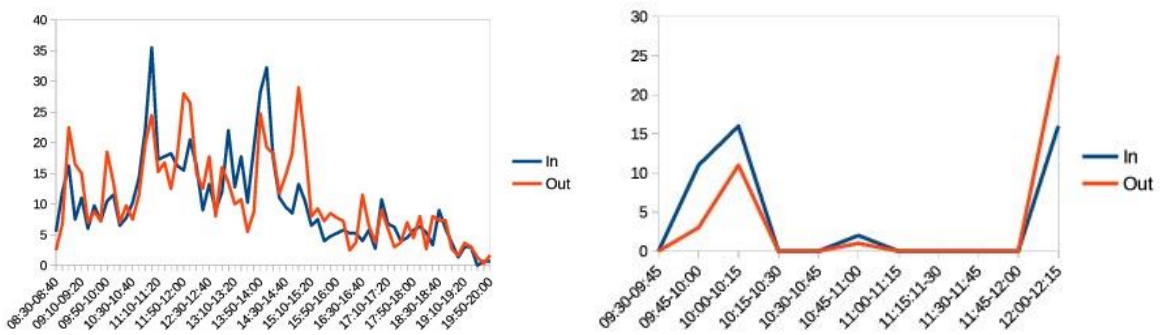



Fig. 3. Arrivals and departures.

Measuring infrastructure consisted on an infrared people counter that took account of the number of people that cross the entrance to the venues and the instant it took place. The device we used (SensMax Pro TCPIP[1]) was able to discern the direction of the movements and therefore, arrivals and departures to the venue were counted separately. Crossing information is completely anonymous, which makes this counting method privacy-respectful and usable in public places. Nevertheless, this lack of information makes it impossible to relate arrivals with their corresponding departures, which complicates data processing. Once the information was gathered, some representative results were chosen to be used in simulation. In the cafeteria case, results show a similar pattern during the 4 days of measurement. As a result, the average of arrivals and departures (Fig. 3(a)) was selected for simulation. In the conferences case, a similar pattern could also be found. However, the different nature of the conferences and the diverse number of attendees led us to choose the result from one of them instead of a combination of data. Fig. 3(b) shows arrivals and departures in that representative one. Next step was to find data probability distributions to be used in simulations, both for inter-arrrival and waiting time. They should match as much as possible times in data gathered.

Table 1. Data for Conference Escenario Simulation (in Minutes)

(a) Inter-arrival Time

| Interval | Distribution | Arrivals |
|---|---|---|
| 9:45-10:10 | E(0.892) | 27 |
| 10:10-12:00 | E (37.039) | 3 |
| 12:00-12:15 | E (0.282) | 15 |

(b) Waiting Time

| Interval | Distribution | Leaving |
|---|---|---|
| 9:45-10:10 | E (1.759) | 13 |
| 10:10-12:00 | E (20.583) | 1 |
| 12:00-12:15 | E (1.759) | 31 |

Table 2. Inter-Arrival Data for Cafeteria Simulation (in Minutes)

| Interval | Type | Arrivals |
|---|---|---|
| 08:30-09:20 | s2 | 34 |
| 09:20-10:40 | s3 | 88 |
| 10:40-11:20 | s1 | 157 |
| 11:20-12:20 | s2 | 37 |
| 12:20-13:00 | s3 | 22 |
| 13:00-13:40 | s2 | 84 |
| 13:40-14:40 | s1 | 140 |
| 14:40-15:20 | s3 | 17 |
| 15:20-17:00 | s4 | 50 |
| 17:00-17:30 | s3 | 24 |
| 17:30-20:00 | s4 | 28 |

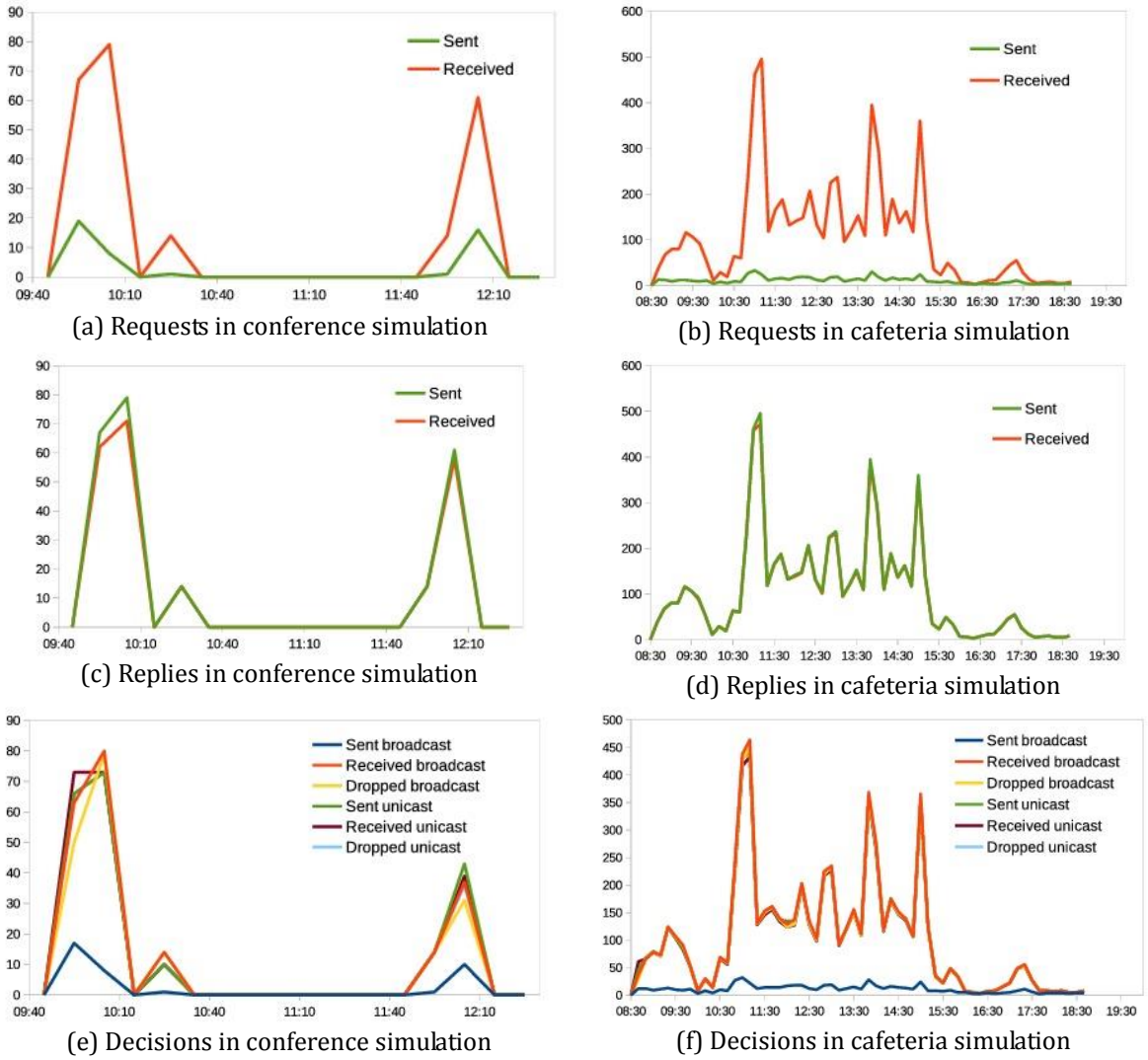
| Type | Distribution |
|---|---|
| s1 | N (0.574,0.337) |
| s2 | N (0.762,0.579) |
| s3 | N (1.197, 1.431) |
| s4 | N (2.486, 0.749) |

As shown in graphs, people movement distribution changes according to time. Change happens in the conferences scenario between start time, actual conference time and end time. In cafeteria, peak times coincide with lunch and mid-morning snack times, around 14h and 11h respectively.

That is to say, it is not possible to find a distribution that matches the whole measurement time for any of the cases. To find out inter-arrival distribution, this time was split into different intervals that were classified according to the people movement pattern found on them. 11 different interval were found in the

[1]www.sensmax.eu

cafeteria data. They were classified in 4 interval types, named from s1 to s4 from the busiest to the quietest category. The number of intervals decreased to 3 in the conference case. A probability distribution was then searched for for every interval, using average inter-arrival time measured. Time spent on the place (modeled as waiting time) was more difficult to be determined as there was no way to relate arrivals and departures and therefore, to assess how long were people actually on the place. As a consequence, probability distribution was approximated. In the cafeteria case, an exponential distribution with mean 9 min was chosen as it was found to suit reasonably the distribution when comparing the measured departure graph. These results can probably be improved by using also a time-based interval division since time spent on the cafeteria is likely to be dependent on the moment of the day, e.g. people spend more time eating lunch than having a coffee. In the conference case a different approach was followed. Waiting time distributions choice was also interval-based but not in the same way as inter-arrival time. Instead of providing nodes with a certain waiting time regarding the time they arrive, they are provided with a waiting time regarding the time they are required to leave.




(a) Requests in conference simulation

(b) Requests in cafeteria simulation

(c) Replies in conference simulation

(d) Replies in cafeteria simulation

(e) Decisions in conference simulation

(f) Decisions in cafeteria simulation

Fig. 4. Messages exchanged in simulations.

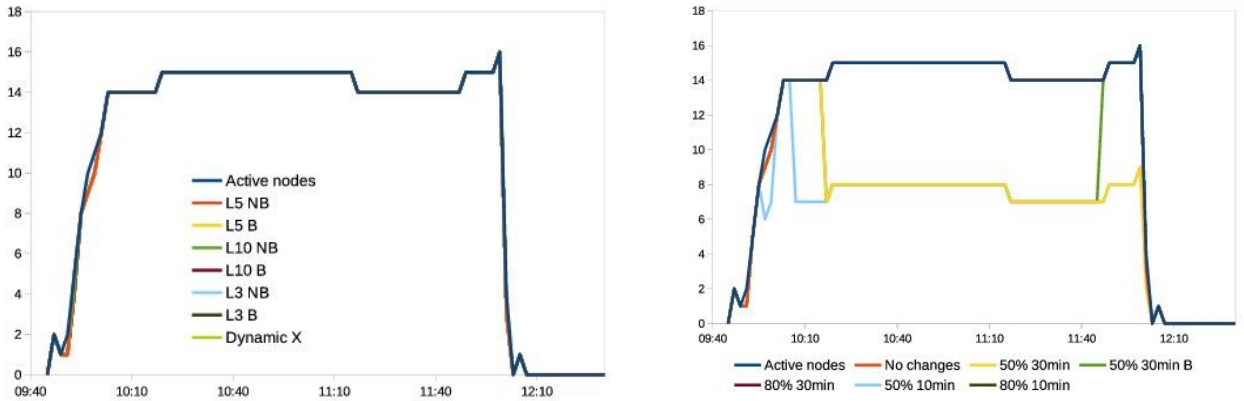
As a result, nodes that belong to a certain arrival interval may belong to a different departure interval. Furthermore, as most nodes start leaving the room at a deterministic time (conference end time), nodes in the third interval add their random waiting time to the end time instant. That is to say, in this case nodes

leave at 12 + waiting time. Waiting time distributions are shown in Table 1(b). Probability distributions determination can probably be improved by using specific algorithms, at least in the inter-arrival times case, where mean values are known. A possible tool could be Matlab function *allfitdist*.

## 6.1. Results

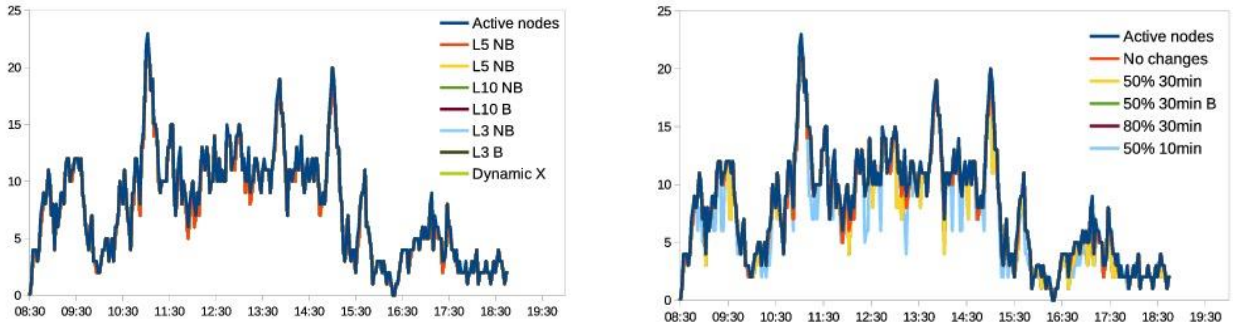
Once parameters modeling was decided, the protocol was implemented and simulation was performed using ns-3 network simulator.

The number of nodes used was chosen as the average of the amount of nodes that were found in every interval in the measurements and are shown in Tables 1(a) and 2. Samples were generated for both arrival and waiting time and events were scheduled for arrivals and departures. Nodes are randomly placed in a 10x10 m area on which they move, also randomly. Result evaluation focuses on network performance and decision contagion. Figure 4 show the number of messages of each type that were sent, grouped every 10 min. They display how packet flow concentrates on the same instants where arrivals take place (Fig. 3). Fig. 4(a) and Fig. 4(b) show no message losses in the request phase. That almost happens in the reply case, although the fact nodes at the same distance will receive requests at the same time and therefore answer simultaneously provokes the collision of some of the replies. This number is low and does not affect the performance of the system. For decision packets, we implemented unicast and broadcast delivery. In both cases, received and dropped packets are almost the same while sent packets significantly increase in the unicast case, as one is created for every nearby node. Decisions that match current state are dropped because it is not possible to learn from them.




(a) Different learning schemes (b) When periodic changes are forced

Fig. 5. Active nodes and number of nodes with the same state in conference simulation.

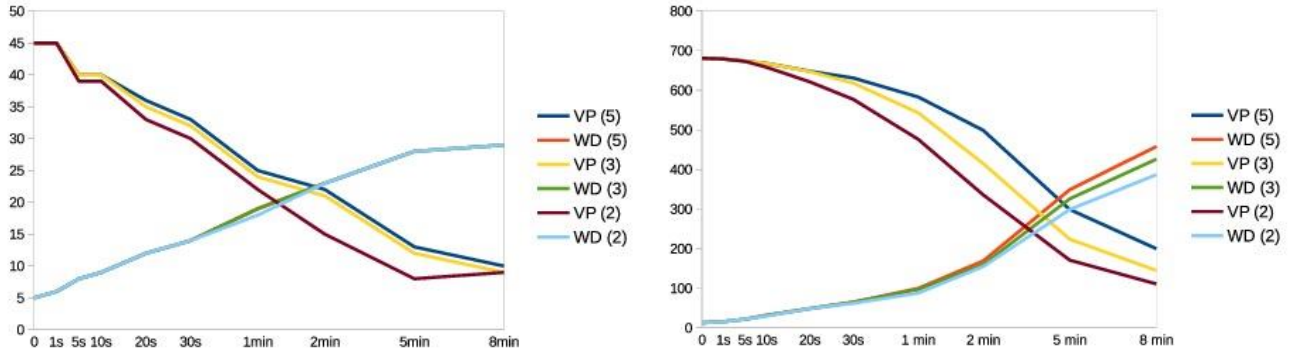



(a) Different learning schemes (b) When periodic changes are forced

Fig. 6. Active nodes and nodes with the same state in cafeteria simulation.

That is the reason that explains the high rate of dropped decisions, both in the broadcast and the unicast

case. Figs. 5 and 6 show the number of nodes that share state. They display how nearly every active node's state agrees in both scenarios, whatever the number of decisions required for learning is (3, 5 or 10) and independently of whether decisions are broadcast after reconsideration (B) or not (NB).




(a) Conference (b) Cafeteria

Fig. 7. Voting processes and nodes that left the area without decision during simulation.

In order to get a deeper insight into the learning performance, periodic state changes were forced. We used different change frequencies and diverse percentages of nodes whose state was changed. Results in Figs. 5(b) and 6(b) also show unanimous state agreement, as long as the percentage of nodes whose state is changed is over 50%. In the cafeteria case performance is better due to the fact arrivals are produced at any time, which provokes more dynamism. Time to receive replies is really short, waiting just 2s is enough in the conference scenario and implies a low number of late replies in the cafeteria one (Table 3). That ensures nodes get replies from nearly every active node in the area in a time interval short enough to allow fast adaptation. As attendants to the conference are normally sat instead of moving around, conference simulation was also performed with static nodes. Due to the already mentioned range of the WiFi technology, no differences were found, as nodes are never moving out of it.

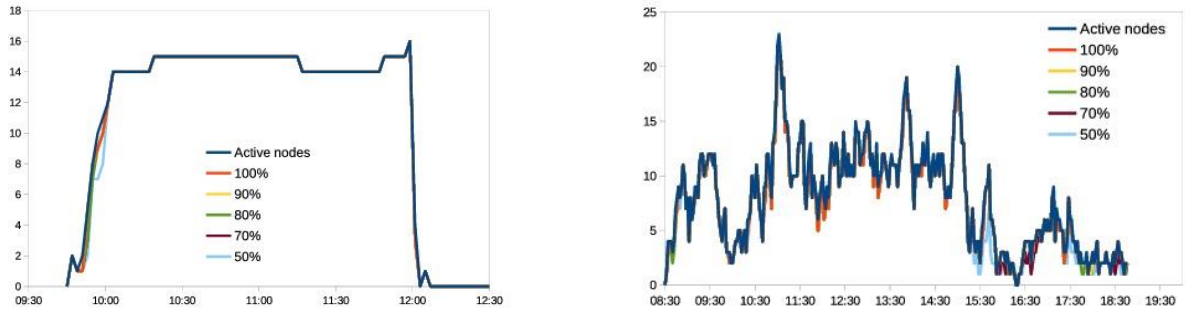
Table 3. Percentage of Replies Received Late in Simulations

| | (a) Conference simulation | | | | | | (b) Cafeteria simulation | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Time nodes wait for replies | 0.5s | 1s | 1.5s | 2s | 5s | | 0.5 s | 1s | 1.5s | 2s | 5s |
| Late replies | 20% | 21% | 0% | 0% | 0% | | 20% | 21% | 1% | 1% | 0% |

## 6.2. Optimization Evaluation

Request replacement with learning has effectively decreased the number of voting processes (shown as VP in Fig. 7). It has also increased the amount of nodes that leave without a decision (shown as WD) and that, therefore, have not adapted their behavior to the circumstances. It could be argued that this goes against the purpose of the system. However, those are nodes who remain in the area for a short period of time so it may not be worthy for them to adapt. That is also the reason why times to wait before request considered are greater than time necessary to make a decision. Smartphones will require a stay duration of at least some minutes to make adaptation worth the effort. Sending broadcast decisions after state changes has been proved to be unnecessary as it represents no improvement in state contagion (Figs. 5 and 6) and increases the amount of sent packages.

Dynamic X calculation does not alter performance but it has been proved to be able to successfully adapt to changing number of nodes (Figs. 5 and 6). It has been already shown how most nodes states virtually always agree. Due to this fact random replies were expected to decrease network load while maintaining performance. Simulation results have confirmed this assumption (Fig. 8).




(a) Conference (b) Cafeteria

Fig. 8. Nodes that share state depending on reply probability.

## 7. Discussion

As far as we know, no context-oriented protocol has been previously designed. Therefore, no direct comparison can be performed. On the one hand, previous collaborative approaches [4] prove the usefulness of cooperation but they do not mention how it can be carried out. The only distributed consensus protocol that was found [14] takes a different approach and verifies its performance in situation detection by comparison with real data. On the other hand, general consensus protocols focus their performance studies on behavior in case of crashed nodes [15], which makes no sense in our high-mobility scenario. Network metrics considered by Alekseish and Ezhilchelvan [15] include number of rounds, time overhead, and packet overhead. Time overhead accounts for decision time, which in our case is fixed by the time nodes wait for answers and therefore, cannot be considered a performance parameter. Number of rounds does not apply either as we only take one per request (a single round trip).

Time to total quiescence may, however, apply for our protocol. In [15], quiescence time decreases with increased density and with nodes speed. However, as this experiment was performed in a 1000×1000 area situation is not alike. Nevertheless, it is unlikely that it could took our protocol hundreds of seconds to reach a quiescent state, whatever number of nodes is chosen.

Also due to the small size of the scenario node speed is not a parameter that affects our protocol.

It should be taken into account that most consensus protocols, such as well-known Paxos [12], [16] and variants [17], are targeted at maintaining replicated files and databases. Therefore, it is logical that they have been designed in a different fashion. They work with a number of members that are given different roles and although there are mechanisms for membership changes, as the group is assumed not to change frequently, they are rather cumbersome. There have been attempts to increase flexibility [18] but they are still not enough for a high-mobility environment such as ours. The role division represents another drawback since it demands a previous establishment of voters' network. Dynamic average consensus approaches [19] take into account cluster changes, divisions and aggregations. However, they assume all the nodes will reach the same conclusion by themselves so they just flood the network with current data and do not propagate any feedback from the nodes. The same idea is used in reputation propagation in [20]. Proposals for consensus with unknown participants [21] study the formal requirements (connectivity, completeness and accuracy of failure detection) and a method to establish a network between the nodes. Randomized protocols form another category of consensus protocol. In those, there is not one correct decision neither a criteria to choose it, any of the possible ones can apply. As a result, the emphasis is on making sure there is only one chosen value and that every node is aware of it [15].

## 8. Conclusion and Future Work

We have discussed the context management problem from a collaborative point of view. Then, we have

proposed a distributed protocol to enable cooperation and fast adaptation to changing circumstances. In order to ensure a feasible implementation, we have only considered technologies already present in current smartphones. We have proved its good operation in two different scenarios, which were simulated using collected real data. The protocol has been shown to be simple and effective at the targeted task, barely leaving room for improvements.

In this first approach, we have not considered other voting criteria than physical neighborhood. However, it may be useful to analyze other factors such as social closeness, the time nodes have spent in the place, previous visits or their reputation from previous voting processes. Our current work focuses on that study, which we plan to apply in scenarios where there is not a clear appropriate behavior everybody should apply. Moreover, we plan to carry out a real-world evaluation to test our protocol operation on different events.

## Acknowledgments

This work is funded by: the European Regional Development Fund (ERDF) and the Galician Regional Government under agreement for funding the Atlantic Research Center for Information and Communication Technologies (AtlantTIC); the Spanish Government and the European Regional Development Fund (ERDF) under project TACTICA; and the Spanish Ministry of Economy and Competitiveness under the National Science Program (TEC2014-54335-C4-3-R).

## References

[1] Maekawa, T., Yamashita, N., & Sakurai, Y. (2014). How well can a user's location privacy preferences be determined without using GPS location data. *IEEE Transactions on Emerging Topics in Computing*, *6750(c)*, 1–13.

[2] Sadeh, N., Hong, J., Cranor, L., *et al*. (Oct. 2008). Understanding and capturing people's privacy policies in a mobile social networking application. *Personal and Ubiquitous Computing*, *13(6),* 401–412.

[3] Bigwood, G., Abdesslem, F., & Henderson, T. (2012). Predicting location-sharing privacy preferences in social network applications. *Proceedings of AwareCast* (pp. 1–12).

[4] Huuskonen, P., Mäntyjärvi, J., & Könönen, V. (2010). Collaborative context recognition for mobile devices. *Handbook of Ambient Intelligence and Smart Environments. Springer US*, 257–280.

[5] Mantyjarvi, J., Himberg, J., & Huuskonen, P. (2003). Collaborative context recognition for handheld devices. *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*.

[6] Bilogrevic, I., Huguenin, K., Agir, B., Jadliwala, M., & Hubaux, J.-P. (2013). Adaptive information-sharing for privacy-aware mobile social networks. *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (p. 657).

[7] Han, J., Schmidtke, H. R., Xie, X., & Woo, W. (Aug. 2014). Adaptive content recommendation for mobile users: Ordering recommendations using a hierarchical context model with granularity. *Pervasive and Mobile Computing*, *13*, 85–98.

[8] Du, J., Chen, W., Liu, Y., Gu, Y., & Liu, H. (2013). Catch you as I can: Indoor localization via ambient sound signature and human behavior. *International Journal of Distributed Sensor Networks*, 1–16.

[9] *Balakrishnan*, M., Mohomed, I., & Ramasubramanian, V. (2009). Where's that phone? Geolocating IP addresses on 3G networks. *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement* (p. 294).

[10] Christin, D., Sánchez, L. P., Reinhardt, A., Hollick, M., & Kauer, M. (2013). Share with strangers: Privacy bubbles as user-centered privacy control for mobile content sharing applications. *Information Security Technical Report*, *17(3),* 105–116.

[11] Consolvo, S., Smith, I., Matthews, T., LaMarca, A., & Powledge, P. (2005). Location disclosure to social relations: Why, when, & what people want to share. *Proceedings of SIGCHI Conference on Human Factors in Computing Systems* (pp. 81–90).

[12] Lamport, L. (1998). The part-time parliament. *ACM Transactions on Computer Systems*, *16(2),* pp. 133–169.

[13] Petit, J., & Mammeri, Z. (2011). Dynamic consensus for secured vehicular ad hoc networks. *Proceedings of IEEE 7th International Conference on Wireless and Mobile Computing, Networking and Communications* (pp. 1–8).

[14] Raumer, D., Fuchs, C., & Groh, G. (2014). Reaching consensus among mobile agents: A distributed protocol for the detection of social situations.

[15] Alekeish, K., & Ezhilchelvan, P. (Mar. 2012). Consensus in sparse, mobile ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, *23(3)*, 467–474.

[16] Lamport, L. Paxos made simple. *ACM Sigact News*, 2001.

[17] Ongaro, D., & Ousterhout, J. (2014). In search of an understandable consensus algorithm. *Proceedings of USENIX Annual Technical Conference*.

[18] Sanderson, D., & Pitt, J. (Sep. 2012). Institutionalised consensus in vehicular networks: Executable specification and empirical validation. *Proceedings of IEEE Sixth International Conference on Self-Adaptive and Self-Organizing Systems Workshops* (pp. 71–76).

[19] Spanos, D., Olfati-Saber, R., & Murray, R. (2005). Dynamic consensus on mobile networks. IFAC World Congress.

[20] Liu, Y., & Yang, Y. R. (2003). Reputation propagation and agreement in mobile ad-hoc networks. *IEEE Wireless Communications and Networking, 3*, 1510–1515.

[21] Cavin, D., Sasson, Y., & Schiper, A. (2004). Consensus with unknown participants or fundamental self-organization. *Ad-Hoc, Mobile, and Wireless Networks*, *8*.

**Fátima Castro-Jul** received the telecommunications engineer degree from the University of Vigo, Spain in 2014. Currently, she works as a researcher in I&C Lab (iclab.det.uvigo.es), at the AtlantTIC research center (atlanttic.uvigo.es) in the Univeristy of Vigo, where she is enrolled in the PhD program in information and communication technologies. Her research interests include ambient intelligence, context awareness, ubiquitous computing and networks.

**Ana Fernández-Vilas** received her Ph.D. degree in computer science from the University of Vigo in 2002. In 1997, she joined the Department of Telematics Engineering (University of Vigo) where today she is an associate professor. She is engaged in social data mining and ubiquitous computing environments, being a member of the Information & Computing Lab (iclab.det.uvigo.es) at the AtlantTIC Research Centre (atlanttic.uvigo.es).

**Rebeca P. Díaz-Redondo** has a Ph.D. degree in computer science, she is an associate professor at the Department of Telematics Engineering and member of the Information & Computing Lab (iclab.det.uvigo.es) at the AtlantTIC Research Centre (atlanttic.uvigo.es) of the University of Vigo. She currently works on applying social mining and data analysis techniques to different areas.