Exploring data forwarding with Bluetooth for participatory crowd monitoring

Christin Groba
Chair of Computer Networks
Technische Universität Dresden
Dresden, Germany
Christin.Groba@tu-dresden.de

Thomas Springer
Chair of Computer Networks
Technische Universität Dresden
Dresden, Germany
Thomas.Springer@tu-dresden.de

***Abstract*—Participatory crowd monitoring estimates the size and dynamics of a crowd based on position data shared by people in the crowd. At large-scale events, these small but frequent uploads compete for bandwidth with the crowd's social networking activities causing transmission errors and long delays. Forwarding collected data within a peer-to-peer network allows for bundling that data and assigning the upload task to a few selected peers. On modern phones, however, peer discovery and networking is challenging due to tight restrictions of mobile operating systems. This paper shows how the announcement of internal state transitions via Bluetooth enables peers to align their operations and to accommodate for issues such as bandwidth restrictions, inexact timing, limited background activity, and idle times. Experiments on Android phones show that already for a small set of phones the proposed protocol works reliably and significantly reduces the use of the networking infrastructure. With the data delay and energy consumption in acceptable bounds, the protocol presents a viable solution for when network overload puts crowd monitoring data at risk of not being delivered in a timely manner.**

***Index Terms*—crowd monitoring, data bundling, peer-to-peer forwarding, Bluetooth, Android**

I. Introduction

Participatory crowd monitoring uses the sensing and networking capabilities of modern phones to collect data about a crowd of people. Participants within the crowd agree to run an app that, for example, anonymously shares their position. Imagine such a monitoring campaign at a large music festival. Analyzing shared position data already during the event allows for creating a live heat map that color-codes the popularity of different stages and concession stands. With exclusive access to that map, participants may adjust their visit based on their level of preferred crowdedness. Event organizers, on the other hand, may use the map to efficiently deploy emergency response personnel and other critical resources. Typically, such campaigns run in a way where each participant uploads position data to remote campaign servers directly. However, at large-scale events the demand for networking bandwidth increases rapidly while existing infrastructure is designed for the average case that does not accommodate for large crowds. With the music festival taking place in a rural area, for example, and visitors immediately sharing their festival experience on social platforms, Internet access becomes slow and at times even impossible [1]. The updates of location data then compete for bandwidth just like any other traffic and may not arrive soon enough for the live visualization of visitor densities.

The upload of bundled data may help to mitigate the stress on the networking infrastructure. For this, phones communicate peer-to-peer (p2p) via an alternative communication channel like Bluetooth to forward data and to select hubs among themselves that upload bundled data to the campaign server. However, such a concept is challenging to implement on modern phones due to restrictions set by mobile operating systems (mobile OS) that limit background activities with respect to when and for how long they are executed and access to networking resources that require explicit user authorization.

This paper presents a state-based protocol for bundling crowd monitoring data, forwarded with Bluetooth, to reduce their use of the networking infrastructure. The approach is grounded in work related to collaborative sensing and device-to-device communication (Section II). It complements existing research by studying the protocol's deployment and performance on modern phones in a real-world operating environment. Challenged by that environment, the paper explores the restrictions of mobile OS that prevents from a straight forward implementation of a forwarding protocol with Bluetooth. It explains how the protocol is designed to handle the challenges related to a phone's limited background activity, bandwidth restrictions, inexact timing, and idle times (Section III). Key to the proposed solution is the announcement of internal state transitions that enables peers to coordinate their discovery and networking actions. This way the protocol ensures that if nearby phones currently run the monitoring app, they exchange and upload the corresponding data successfully. Experiments with up to five Android phones quantify the protocol's effectiveness, reliability, and overhead (Section IV). In addition, first insights from a field trial regarding user acceptance are included. The paper ends with a discussion of the protocol's performance results (Section V).

The contribution of this paper is its hands-on approach to bundling and data forwarding among modern phones, highlighting how ever tighter restrictions of mobile OS impact the design of protocols for participatory crowd monitoring.

This research is funded by the German Research Foundation (DFG) under GR 4517/1-1.

II. Related work

The problem of coordinating modern phones in a p2p manner has been studied in the area of collaborative sensing and device-to-device communication.

Collaborative sensing explores how nearby devices may cooperate on energy-intensive tasks like GPS-based positioning or audio recording. The challenge is to strike a balance between the energy required for sensing and the availability and accuracy of the measurements. One way to achieve this is by comparing samples from different sources and deactivating those devices that provide redundant data [2]. In the example of GPS-based positioning, some devices sample and share their measurements while other devices do not sample themselves. They rather wait and calculate their position based on their peers' measurements [3]. Solutions with a central entity to make such a decision [2], [3] have full system view to run the selection algorithm. However, they assume stable connectivity between the central entity and the devices. This may not always be the case in crowded scenarios.

A distributed approach [4] lets devices broadcast their measurement, energy budget, and time of next sensing. This way co-located devices decide whether to wait or to sample and share. This solution is applicable, if devices have full control over when to execute an action. However, mobile OS restrict this ability, in particular for background activities. Rather than exact execution times, they only allow for specifying execution windows. This introduces uncertainty during the decision making.

Another distributed solution [5] lets mobile nodes decide for themselves whether to become a hub for collecting sensor data from peers. A stochastic algorithm makes this decision in intervals and ensures a fair and effective allocation. It assumes that all nodes have the same understanding of when an interval starts and ends. This is not the case for modern phones which operate on individual time windows. As a result, a node may decide to resign from its role as a hub while peers still send data to that node.

The protocol in this paper addresses this inexact timing issue and enables phones to align their actions even when they operate in different protocol states. Further, the experiments run on Android phones that actually exchange sensor data via their Bluetooth interface and allow for performance tests under real-world conditions. This goes beyond simulation-based [5] and trace-driven evaluations [4].

Device-to-device communication among mobile phones requires coordination to select the device that manages a group of peers and relays their messages. In WiFi Direct this role of a group owner needs to be dynamically re-assigned to share the cost of increased energy consumption among all group members. A study [6] found that among different group owner selection algorithms, random self-selection is most suitable for large and highly dynamic networks. In intervals a group owner disconnects from its members and waits for invitations to another group. If those do not arrive, the former group owner re-assumes its role and sends out invitations to its own group. For inter-group communication, so-called traveling phones autonomously disconnect from their current group and connect to another group to disseminate content between groups [7]. The decision whether to become a traveling node is also based on self-selection and a stochastic component that takes group cardinality into account.

In WLAN tethering, some devices change into access point (AP) mode and relay messages while other devices scan for access points and connect as clients. However, phones in the same mode are not aware of each other and some intelligent switching between modes is required. One way is to achieve this is to predict the dissemination process and maximize the probability of two random phones to connect [8]. Another way is to randomize the time a phone remains in a particular mode [9]. Replacing the required but energy-intensive Wifi scans with a discovery based on Bluetooth Low Energy does not only reduce the energy consmuption of the AP but also creates larger groups with less redundant APs [10].

In this paper, phones compare the remaining battery level of their peers to decide where to forward local sensor data. This way the role of a hub that uploads all data rotates naturally depending on the availability and characteristics of the peers in vicinity. As for the communication among the phones, the choice for classic Bluetooth is due to its widespread availability on Android phones and its unrestricted use in conjunction with regular Internet access. In comparison, in WiFi Direct the support for simultaneous connections to a group and a regular WiFi AP is optional. This means, devices without that feature cannot use their regular Internet connection while sharing data via WiFi Direct. With WLAN tethering, on the other hand, a phone in AP mode drops the connection to any regular WiFi AP and instead connects to the cellular network. This may incur monetary cost and discourages participation. Bluetooth Low Energy (BLE) is another networking option and typically used for phones to communicate with low energy peripherals like body sensors. If phones want to communicate with each other, they need to be discoverable like a peripheral. Support for peripheral mode, however, depends on the firmware and varies even for newer Android phones.

III. P2P data forwarding

The concept of p2p data forwarding is based on the following system setup. We assume a set of nodes representing people with modern phones that run an app to participate in a crowd monitoring campaign. They arbitrarily move around while visiting an event, e.g., a fair or a music festival (Figure 1). Each node has a certain capacity to store a number of position updates and can perform the following tasks:

- **Discover:** Updating the phone's view of available peers by periodically running a discovery,
- **Accept:** Accepting connection requests and storing position data received from peers,
- **Forward:** Forwarding all data saved locally to a selected peer, and

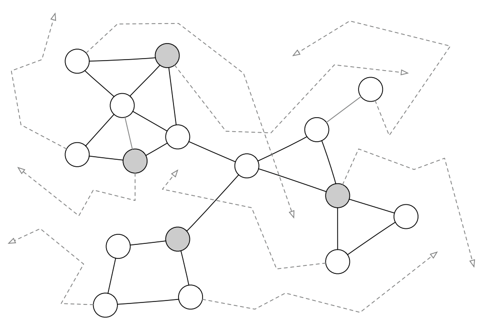

Fig. 1. System setup

- **Upload:** Uploading data locally stored if a connection to the crowd monitoring server is available.

Due to the density of devices we assume that only a subset of all nodes is able to upload position data directly to the server based on WiFi or cellular (marked as grey filled circles in Figure 1) while all other nodes have to rely on data forwarding.

A. Restrictions by mobile OS

A set of restriction imposed by mobile OS have a significant impact on the design of a forwarding protocol with Bluetooth. The following requirements were collected during a set of previous iterations of the presented protocol:

Limited background activity: Crowd monitoring requires upload of position updates even if the user is not actively interacting with the app. However, mobile OS save resources and energy by limiting the execution of background processes. As soon the user stops interacting with an app, the app goes into the background mode, that is only long enough to run some clean-up tasks, before the system inactivates the app. An inactive app wakes up only few times an hour to run its background processes with a short time window. This reduces the availability of peers significantly.

Bandwidth restrictions: Conceptually, a phone may discover and establish connections to peers simultaneously. In reality, however, these actions compete for bandwidth, slowing down connection attempts and reducing the quality of existing connections. Phones must therefore switch between either discovering or accepting connection requests and forwarding data. This implies that two phones cannot exchange data as long

Inexact timing: Sensing campaigns define intervals as to when data should be collected and forwarded but the mobile OS batches background processes and prevents them from running exactly at a scheduled time. This inexact timing means that time-based schedules for coordinating discovery and networking between peers are impractical as the app's control over when a task gets executed is limited.

Idle times: During idle times, which result from periodic upload activities in the sensing campaign, the phone needs to convey to its peers that it is not available for interaction to conserve system resources. Turning off the beaconing that enables peers to discover the phone, however, is not an option since the user needs to manually authorize each such change.

B. Resulting requirements for protocol design

Conceptually, the identified tasks run in parallel but due to the phone's *bandwidth restrictions* discovery and networking need to take separate phases. Due to the *inexact timing restriction* switching between these phases should not rely on exact timeouts. Further, two phones need to coordinate their tasks since they cannot exchange data while they both run the discovery. Therefore, random start-up times should be used in the protocol, but with discovery taking time (for Bluetooth in Android 12 seconds), the lower the interval for the protocol to run, the more likely that discovery phases overlap.

Also, with *limited background activity*, the protocol needs to ensure that if co-located phones happen to execute the monitoring app at the same time (possibly as a background process), they collectively work towards a successful data exchange and upload. Finally, the protocol needs to convey *idle times* in-between protocol runs for when the phone is not available for accepting and forwarding monitoring data. Changes to the phone's discoverability requires the user's consent and cannot be requested on a regular basis without annoying the user.

C. State-based protocol

A phone runs the protocol in intervals triggered by a timeout. The protocol consists of a server-part and a client-part which run simultaneously on the phone (cp. Figure 2).

The **server-part** starts in ”discovering” state and waits for the client-part to do the same (or to report being finished for this interval). When this happens, both parts are synchronized and ready to jointly run the discovery (1). If the client-part prematurely cancels discovery, the server-part starts over (2). As soon as the server-part detects a peer whose client-part is discovering, it cancels local discovery and goes into ”accepting” since it anticipates the peer's connection request (3). Otherwise, discovery eventually times out and the server-part finishes (4). Once the time for accepting connection requests runs out, the server evaluates whether it has been server long enough in this interval and finishes (5) or otherwise gets ready to run discovery again (6).

The **client-part**, like the server-part, starts in ”discovering” and waits to run the joint discovery (7). If discovery gets cancelled by the server-part, the client-part starts over (8). As soon as it discovers a peer with an accepting server-part, it cancels discovery and forwards all its data (9). If discovery times out and no such peer has been discovered, the client-part directly uploads its data to the remote campaign server (10). If discovery times out and a peer with its server-part in ”discovering” state is around (11), the client-part evaluates whether to run discovery again (12), since the peer is likely to go to ”accepting” state soon. If the client-part is unable to forward or upload its data within a maximum number of rounds, it finishes by saving its data locally and waiting for the protocol to be triggered again (13).

The protocol achieves coordination between phones by including the state of the server-part and the client-part in the discovery beacons. This way a phone is aware of its

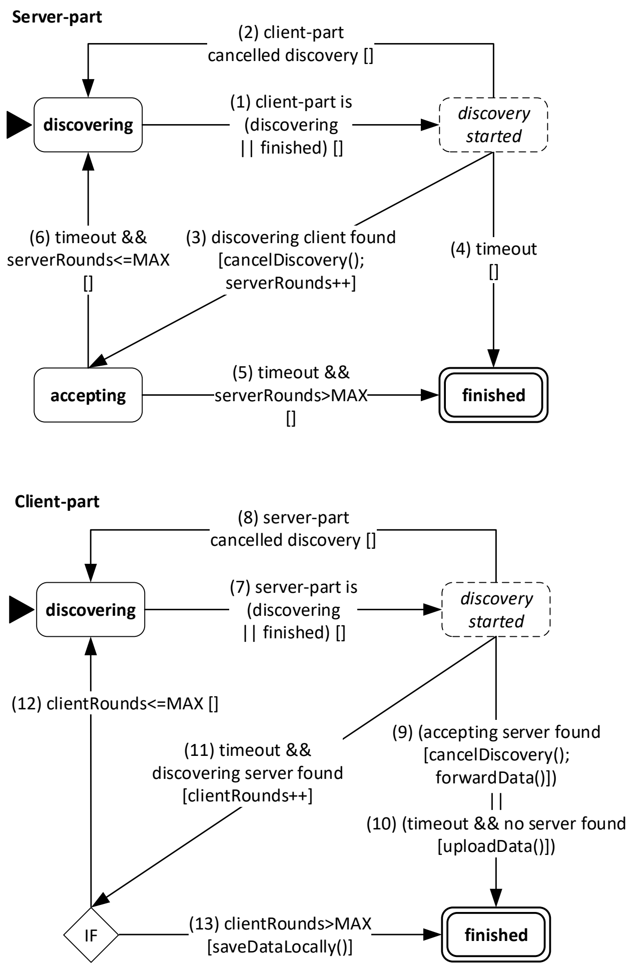
Server-part
(2) client-part cancelled discovery []
discovering
(1) client-part is (discovering || finished) []
discovery started
(6) timeout && serverRounds<=MAX []
(3) discovering client found [cancelDiscovery(); serverRounds++]
(4) timeout []
accepting
(5) timeout && serverRounds>MAX []
finished
Client-part
(8) server-part cancelled discovery []
discovering
(7) server-part is (discovering || finished) []
discovery started
(12) clientRounds<=MAX []
(11) timeout && discovering server found [clientRounds++]
(9) (accepting server found [cancelDiscovery(); forwardData()]) || (10) (timeout && no server found [uploadData()])
IF
(13) clientRounds>MAX [saveDataLocally()]
finished

Fig. 2. Protocol to run on a phone. Bold-face states are visible to peers while an italic-face state is only visible to the phone itself. State transitions read as *trigger[list of actions]*.

peers, can cancel discovery deliberately, and proceed with the networking tasks. This also ensures, that phones that currently run the protocol, interact. In particular, the client-part stays active as long as discovery suggests that potential server peers are around. The server-part, on the other hand, stays available as long as it finds active clients. Staying active, however, is configured to a maximum number of rounds to balance the phone's participation commitment with the depletion of its local resources. In terms of conveying idle times, a phone only interacts with peers whose relevant server or client part is unfinished and ignores all other peers.

For selecting the destination, the phone compares the remaining battery level of the discovered peers including its own and identifies the phone with the highest remaining level. If on par, it selects the phone with the highest phone id. The selected destination may be another peer or the phone itself. In the former case, the phone forwards its data to the other peer. In the latter case, the phone uploads its data to the campaign server because among all its peers it is the one with the highest remaining battery level. This partial order avoids forwarding loops. Thresholds for a minimum number of data in the local data pool or a maximum delay time are currently not part of the protocol. Once data forwarding is finished, the phone discards the connection. This is to adapt to the mobility of the phone users and avoids failure due to stale connection information.

D. Implementation details

The choice of Bluetooth comes with three implications on the implementation in Android: First, Bluetooth discovery beacons as such are not configurable to transmit custom data. As a workaround, the protocol edits the phone's device name, which is part of a beacon, to convey the remaining battery level and the state of the client-part and the server-part.

Second, preliminary tests showed that peers are likely to operate on stale state information, because beacons arrive only after some delay at other peers. The protocol addresses this issue by delaying the corresponding action after a phone transitioned between states. This allows for state changes to "sink-in" with its peers.

Third, secure connections to new Bluetooth devices require manual pairing which is not viable in the context of mass events where peers are numerous and change frequently. The protocol uses insecure connections that do not require such user authorization. This means, however, that data is clear text and prone to eavesdropping. It requires security measures on higher network layers similar to SSL where participants encrypt and sign their data such that only the server is able to decrypt each contribution. Such measures remain to be implemented in the protocol.

IV. Experiments

The goal of the experiments is to quantify the protocol's effectiveness, reliability, and overhead in comparison to a non-cooperative baseline where each phone uploads its data directly. Position data are sightings from a iBeacon which will be part of an indoor positioning and crowd monitoring system in future. The setup is static in that neither the beacon nor the phones move and the phones stay in each others' transmissions range. To test the protocol, Wifi is disabled for all but one of the phones. Started randomly, each phone repeatedly runs the protocol or baseline over a measurement period of 30 minutes. The phone triggers the next protocol run 10 to 30 seconds after the previous run finished. An experiment is repeated at least three times to calculate averages and to verify that observed behavior is representative.

A. Network load

Evaluating the protocol's effectiveness, Figure 3 depicts the total number of uploads that co-located phones created over the measurement period. For the baseline, this number increases as each additional phone contributes its own number of uploads which is on average 58 (± 3) uploads. Each such upload contains on average two kilobyte of data. For the protocol, the number of uploads is much lower and decreases towards an average of 10 (± 1) uploads for five phones. Then the protocol-based uploads contain on average 43 Kilobyte

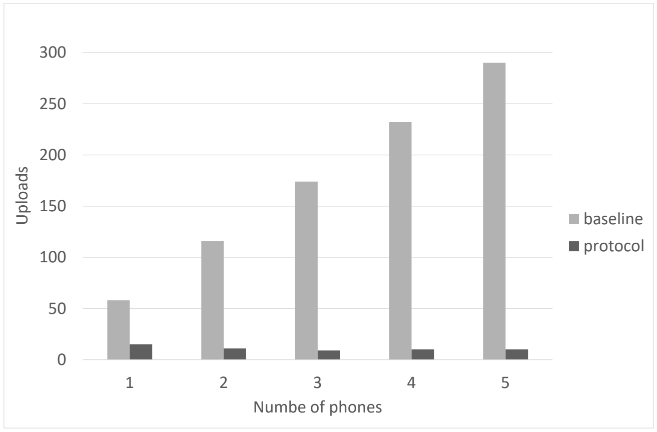

Fig. 3. The protocol reduces the total number of uploads.

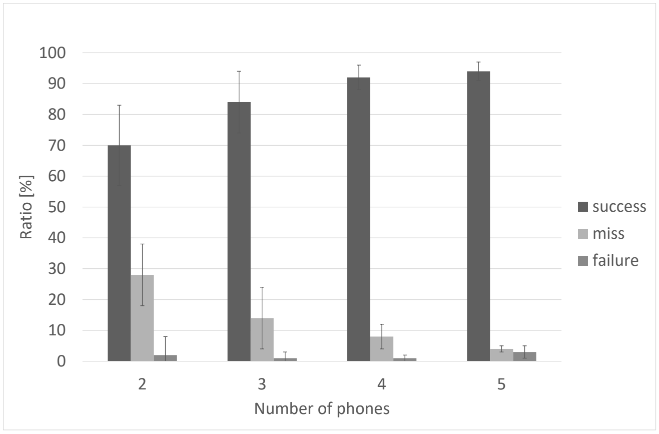

Fig. 4. The protocol allows for reliable p2p data forwarding.

of data. This shows that the protocol meets its objective and reduces the strain on the Wifi network by collecting the bundle of data via the p2p connections first.

B. P2P performance

To assess the protocol's reliability, consider how the client-part finished a protocol run in the experiments: Either, it forwarded data successfully to one of its peers (success). Or, some failure occurred during the forward and the client saved that data locally to try again in the next run (failure). Or, the client missed the opportunity to forward data because each time it run a discovery all potential server peers had already finished their run (miss). Figure 4 depicts the count of these outcomes relative to the overall number of protocol runs as success, failure, and missed ratio. At least two phones had to be part of the experiment to allow for a p2p connection. The average success ratio increases from 70 percent for two phones to 94 percent for five phones. At the same time the probability that a client misses co-located servers decreases from 28 percent to three percent. This is remarkable since the phones are not synchronized and need to coordinate their discovery and networking times. The failure ratio is below three percent irrespective of the number of phones. Failures do not increase despite the increase of phones which increase the number of forwards and the possibility of error. Failure occurred because in some cases the status had been incorrectly set in the implementation while in other cases the delay between changing the status and executing the corresponding action was too short for peers to notice.

C. Data delay

One aspect of the protocol's overhead is how long data takes to be available at the campaign server. Figure 5 shows the delay from sensor data being collected to the time it arrives at the campaign server. For the baseline, data is available within 21 (± 7) seconds independent from the number of phones since each phone uploads data directly. Notice, this result refers to ideal networking conditions of an lab environment and omits

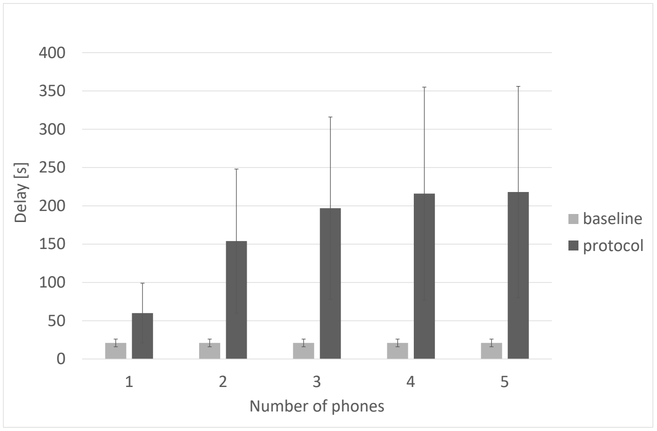

Fig. 5. The protocol delays the availability of data.

the rare cases when the phones experienced connection problems. For the protocol, the delay increases with the increase of co-located phones toward 4 (± 2) minutes. Experiments with a single phone show that there is basic delay due to the time-intensive Bluetooth discovery and Bluetooth server socket. Follow-up experiments with four phones showed that reducing the socket timeout from 30 to 15 seconds, decreases the delay by one minute and increases the failure ratio from 3 to 5 percent. The main delay stems from the active use of the p2p network. In the experiments with 5 phones, data took one hop (in 52 percent of the cases), two hops (40 percent of the cases) and more than two hops (8 percent of the cases) to arrive at the server, at each hop waiting for the peer to make its forwarding decision.

D. Energy consumption

Another aspect of the protocol's overhead is its impact on battery. We recorded the remaining battery level over time for a client peer and a server peer. The client peer was configured to only forward its own data while the server peer only accepted peer data and uploaded to the server. Pretests confirmed that these two represent the lower and upper bound for energy consumption i.e. the overhead of the other phones

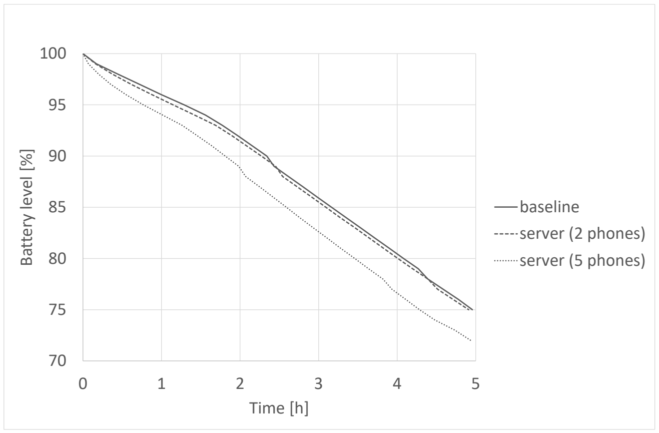

Fig. 6. The protocol's impact on battery is moderate.

that both accept and forward data is somewhere in between these bounds. Figure 6 shows the decline of the phones' battery level for the server peer representing the upper bound. In the two-phone scenario the decline is similar to that of the baseline. This changes with the increase of co-located phones. After four hours of running the 5-phone scenario, the server peer has four percent less battery than the baseline. The measurements for the client peer (not depicted) are similar in that there is no significant difference compared to the baseline if two phones were involved. In case of five phones, the difference after four hours to the baseline was, however, only two percent.

E. User acceptance

In addition to these experiments, we ran a field trial at an indoor fair to study how acceptable it is for visitors to enable Bluetooth, run the protocol and share their positions. For this, the protocol was integrated in the fair's gamified event app with game mechanics for motivating users to activate Bluetooth and share their location. The analysis shows that 36 percent of the app users (in total 27 people) participated and run the protocol for an average duration of 55 minutes. This motivates studying the protocol's performance at larger scale in future.

V. Discussion and Conclusion

This paper addresses mobile crowd monitoring at large-scale events and the problem of long delays for data to become available due to network overload. The proposed protocol collects sensor data from contributors in a Bluetooth p2p network and assigns a selected few that upload the data to the final destination. Working with modern phones, however, means to coordinate the phases for discovery and networking. Global schedules and exact timeouts are not suitable because the phones operate on individual time windows and the mobile OS (rather than the monitoring app) decides when to run a background process. The protocol makes peers aware of their neighbours' state and enables them to align their actions accordingly.

Experiments show that p2p forwarding works reliably already for a small set of co-located phones and reduces the use of Wifi significantly. The lesson learned in terms of Bluetooth communication is that: a) discovery should be cancelled as soon as a suitable communication partner appears and b) that internal state transition should be delayed after their announcement to reduce transmission failure due to stale state information. Further in terms of overhead, the impact on battery is acceptable while the data delay is considerable compared to almost instant availability when phones upload their data directly. This means that if the networking infrastructure is stable, direct uploads will outperform a p2p approach. If, however, networking problems occur and data is at risk of not being shared or being delayed for hours, the protocol is be a viable alternative. The lesson learned here is, the more hops data travels in the p2p network, the lower the impact on the networking infrastructure but the longer for data to arrive at the server.

As the proposed protocol is only one way to forward crowd monitoring data, the next step is to explore other criteria for the forwarding decision and how they improve the balance between network load and data delay.

Acknowledgment

Many thanks to Martin Johst for helping implement the protocol and conduct the experiments.

References

[1] P. Castagno, V. Mancuso, M. Sereno, and M. A. Marsan, "Why your smartphone doesn't work in very crowded environments," in *IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, June 2017, pp. 1–9.

[2] L. Castro, J. Beltrán, M. Perez, E. Quintana, J. Favela, E. Chávez, M. Rodriguez, and R. Navarro, "Collaborative opportunistic sensing with mobile phones," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, ser. UbiComp '14 Adjunct. ACM, 2014, pp. 1265–1272.

[3] T. Xi, W. Wang, E. C. . Ngai, Z. Song, Y. Tian, and X. Gong, "Energy-efficient collaborative localization for participatory sensing system," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[4] J. Eberle, Z. Yan, and K. Aberer, "Energy-efficient opportunistic collaborative sensing," in *IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems*, 2013, pp. 374–378.

[5] R. Loomba, R. de Frein, and B. Jennings, "Selecting energy efficient cluster-head trajectories for collaborative mobile sensing," in *IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–7.

[6] U. Demir, C. Tapparello, and W. Heinzelman, "Maintaining connectivity in ad hoc networks through wifi direct," in *IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2017, pp. 308–312.

[7] V. Arnaboldi, M. G. G. Campana, and F. Delmastro, "Context-aware configuration and management of wifi direct groups for real opportunistic networks," in *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2017, pp. 266–274.

[8] E. Wang, Y. Yang, J. Wu, and W. Liu, "Phone-to-phone communication utilizing wifi hotspot in energy-constrained pocket switched networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 8578–8590, 2016.

[9] S. Trifunovic, M. Kurant, K. A. Hummel, and F. Legendre, "Wlan-opp: Ad-hoc-less opportunistic networking on smartphones," *Ad Hoc Networks*, vol. 25, pp. 346 – 358, 2015.

[10] S. Bergemann, J. Friedrich, and C. Lindemann, "Neighborhood-aware opportunistic networking on smartphones," in *IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2017, pp. 126–134.