Peer-to-Peer Transfers for Crowd Monitoring - A Reality Check

Christin Groba(✉) and Alexander Schill(✉)

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

Abstract. Peer-to-peer transfers allow for sharing crowd monitoring data despite the loss of network connectivity. However, limited insight into real-world deployment contexts can let the protocol design go astray - particularly, if a certain nature of participant behaviour and connectivity changes is assumed. This paper focuses on the delivery of crowd monitoring data. It puts a protocol out for a reality check that switches to peer-to-peer (p2p) communication when the infrastructure network connection is lost. The evaluation at an annual indoor fair asked visitors to make their phones visible to peers, run the protocol, and share crowd monitoring data. The results show that most of the participants formed a large radio cluster throughout the event. This made p2p networking only possible and enabled a more robust upload of crowd monitoring data. However, dynamic switching between infrastructure network and p2p communication also increased the volatility of the system, calling for future optimizations. The presented measurement results provide further insights into these details.

Keywords: Crowd monitoring · Peer-to-peer · Bluetooth · Android · Real-world evaluation

1 Introduction

Awareness of how event visitors roam the venue allows for aligning safety measures and for optimizing the event setup. Monitoring techniques that engage people in the crowd to share sensor data from their phones is a novel way to infer crowd metrics without expensive camera deployments or crowd stewards.

However, due to the crowd density, the demand for networking bandwidth increases while the infrastructure is designed for the average case that does not accommodate for crowds of people. Crowd monitoring data then competes for bandwidth just like any other traffic and participants of the crowd monitoring campaign may not be able to share their data with the campaign server. Off-loading to peers that still have a connection allows for sharing such data nonetheless. Our previous work shows that going as far as using peer-to-peer

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

P. Gaj et al. (Eds.): CN 2020, CCIS 1231, pp. 64–76, 2020.
https://doi.org/10.1007/978-3-030-50719-0_6

Fig. 1. An annual indoor fair is the setting for testing the protocol.

(p2p) data forwarding on default incurs long data delays and therefore infrastructure network connectivity should be used if available [4].

This paper investigates a protocol that switches to p2p data transfer only when a device lost network connectivity. The device then attempts to offload its data to a one-hop peer that is still connected to the infrastructure. During protocol design, however, one makes assumptions (at times unconsciously) about the way participants behave or the network connectivity changes. This raises the question of how the protocol performs in a real-world deployment where participation, user behaviour, and network infrastructure are beyond the designer's control. The evaluation of the protocol is set during an annual indoor fair in one of the buildings on university campus (cp. Fig. 1). Over the course of six and a half hours 47 visitors participated and shared location, connectivity and peer-related data.

The contribution of this paper is an in-depth analysis of the challenges a real-world crowd monitoring scenario has on the design of a p2p data transfer protocol. The paper explores the impact of participation patterns, clusters of visitors, and implications of unexpected network behaviour.

2 Related Work

Participatory crowd monitoring relies on people in the crowd to share crowd monitoring data with a campaign server. GPS readings, acceleration data, or Wifi fingerprints allow for inferring the crowd's density and flow [9] as well as groups of people that move as a cohesive whole [6]. Experience from large-scale

events show that network access via the existing infrastructure becomes slow and at times even impossible [2]. The resulting long delays for crowd monitoring data call for offloading solutions to reduce the load on the network [1]. In our case, participants experienced a high rate of connectivity changes with networking interfaces frequently switching between connected and disconnected. The paper shows, how such an unexpected network behaviour affected the tested protocol.

One way to reduce network load is to establish the crowd metric among all peers and assign one peer that uploads the metric to the server. UrbanCount [3], for example, proposes a distributed crowd counting technique. It builds on an epidemic model where nodes receive radio signals from other nodes and broadcast a list of "seen" nodes. Trace-driven simulations show that such an approach produces a precise count when the crowd is dense. A similar approach [5] based on audio tones shows high scalability and accuracy at much less energy consumption compared to radio-based solutions. As for peer assignment strategies, techniques from collaborative sensing may be adapted that lets mobile nodes decide for themselves whether to become a peer with additional responsibilities [8]. A stochastic algorithm makes this decision in intervals and ensures a fair and effective allocation.

This paper takes another approach by offloading data to a peer only when connectivity issues occur. However, instead of advancing to complex solutions as proposed for domains other than crowd monitoring [11], it first focuses on early evaluation in a real-world deployment.

3 P2P Protocol

The protocol is based on the assumption that a set of nodes representing people with modern phones run an app to participate in a crowd monitoring campaign. They arbitrarily move around while visiting an event, for example, a fair. Each node has a certain capacity to store a number of sensor data and to perform the following tasks:

- Discover available peers in vicinity and make itself visible to other peers,
- Accept connection requests and store data received from peers, and
- Upload data, if a connection to the crowd monitoring server is available.

Further, it is assumed that some nodes experience network disconnects due to the density of nodes. These nodes rely on transferring their data to peers that are able to upload directly to the server via their WiFi or cellular interface.

When the protocol notices a connectivity change event, it switches depending on the change to either the p2p part or the direct upload part (cp. Fig. 2). The p2p part of the protocol starts the peer discovery and stops it as soon as it finds a one-hop peer that is connected. This spares a selection phase, which, when peer availability changes in the meantime, leads to peer connection issues. The node transfers its data and removes its local copy once the transfer is completed. If the connection is closed prematurely, the transfer is aborted, and the data remains at the node. Further, if no connected peer is found, discovery times out and data

```
onConnectivityChange:
  if connection lost then
    directUpload.stop()
    peerTransfer.start()
  else
    peerTransfer.stop()
    directUpload.start()
  end
```

```
peerTransfer.start():
  wait for directUpload to stop
  while !self.stopped do
    start peer discovery
    onConnectedPeerFound:
      stop peer discovery
      transfer data to peer
    onPeerDiscoveryTimeout:
      save data
    sleep(timeout)
  end

peerTransfer.stop():
  self.stopped=true
  stop peer discovery
```

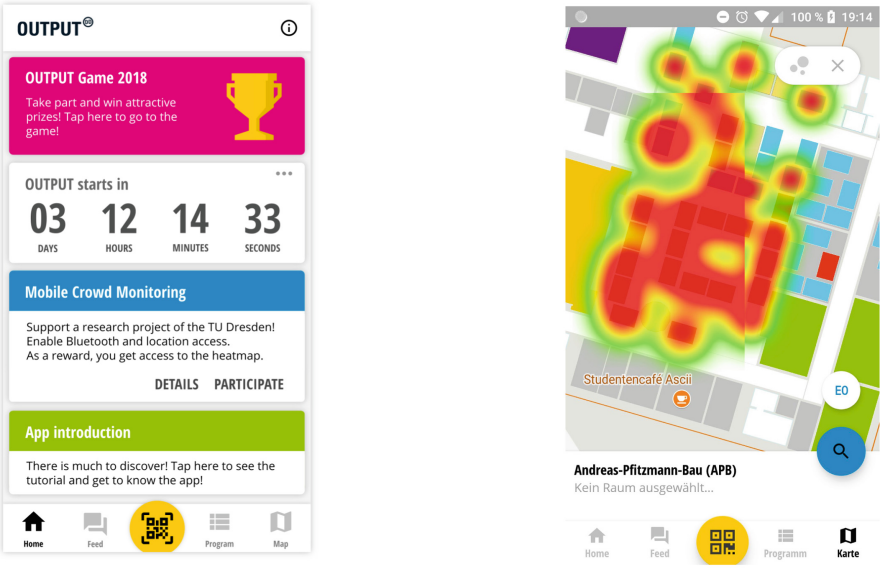
```
directUpload.start():
  wait for peerTransfer to stop
  open server socket
  while !self.stopped do
    upload data
    sleep(timeout)
  end

directUpload.stop():
  self.stopped=true
  close server socket
```

Fig. 2. Protocol pseudocode

stays stored locally. Afterwards, the node idles for a given timeout. The direct upload part of the protocol opens a server socket and in intervals uploads data collected locally and received from peers directly via the infrastructure. When this part is stopped, the server socket closes.

The protocol is in either of two states: peer transfer or direct upload. Given that a state transition is triggered solely by a connectivity change i.e., connected or disconnected, the protocol is considered deterministic. That trigger, however, is managed outside the protocol and may be seen as a shared resource. Locally the protocol is nonetheless deadlock-free, as it remains fully active in its current state until it receives a change notification. From a distributed perspective, the protocol is also deadlock-free, since none of the peers waits for another peer to take action. In case connection requests are not granted or messages do not get delivered, the underlying communication protocol indicates this failure and the node is free to abort or retry communication, possibly with another peer. Timeouts ensure that a node makes progress within a state such as terminating discovery if no adequate peer is found or stop idling to upload data again. With regard to testing, preliminary tests in the laboratory and with a small number of students ensured that the implementation of the protocol is bug-free and meets the research objectives.

The communication among phones is implemented with classic Bluetooth because it is widely available on Android phones and can be used without restricting the regular Internet access. In terms of integrating iOS devices and allowing for cross-platform communication, we also implemented the protocol with Bluetooth Low Energy. While lab tests show promising results, the evaluation in a real-world setting is still on-going. Meanwhile, this paper shares experiences with classic Bluetooth. Phones need to authorize their Bluetooth visibility to become discoverable by peers. Bluetooth discovery, however, cannot be configured to transmit protocol-specific data, namely the connectivity status of the phone. As a workaround, the phone's Bluetooth name is edited,



(a) (b)

Fig. 3. The app accompanying the annual fair OUTPUT.DD provides an opt-in feature for participating in the mobile crowd monitoring experiment and evaluating the p2p protocol (a). Participants have exclusive access to a heat map that color-codes visitor densities in different parts of the event area (b).

which as part of the Bluetooth discovery beacon conveys this information. Further, the Bluetooth-based p2p communication runs over insecure connections to avoid manual pairing of phones. A security protocol like Transport Layer Security (TSL) is necessary to allow for privacy and data integrity despite insecure p2p communication. The implementation of such security measures is still future work.

4 Experiment

The experiment ran during an indoor fair in one of the main buildings on university campus. The exhibition area included an open space of 800 square meters and a number of show rooms spread across three floors. The companion app of that annual event included a opt-in feature for participating in the mobile crowd monitoring experiment and evaluating the p2p protocol (cp. Fig. 3a). The app links program items to an interactive floor plan and applies gamification to increase participation as well as the overall visitor engagement during the fair. Once enabled and all necessary permissions provided, participants shared sensor and log data via their phone's wireless communication interface with an application server in the Internet.

Modern smartphones provide the necessary resources to establish p2p connections, buffer and upload data. Participation, however, draws from the phone's mobile data plan, if Wifi access is unavailable, and consumes scarce battery resources. For latest phone generations like the Google Pixel 3, the battery level drops by two percent per hour running the protocol. More dated phones like the Nexus 5X require three percent per hour. In return for these participation costs, participants had exclusive access to a heat map, which turned the collected data into a map overlay color-coding visitor densities in different parts of the event area (cp. Fig. 3b).

Over the course of the event and among the visitors, about 89 people used the companion app and of those 47 participated in the experiment. The participants shared four sets of data: First, mock sensor readings that the phone created once a second as payload for the protocol. Second, sightings of Bluetooth Low Energy (BLE) beacons that were deployed on site to track the position of the participants. Third, sightings of peers in proximity, which contained the peer's device id, connectivity state, and protocol state. Fourth, the result of each protocol iteration, e.g., whether a p2p transfer was completed or aborted. All data was timestamped on creation and timestamped again when it arrived at the server, running the network time protocol for synchronized clocks. It was buffered locally and uploaded when the device was connected, or as in the case of the mock readings, when a p2p data transfer was possible. In a realistic scenario, the mock readings would be replaced by peer and static beacon sightings to allow for a cluster analysis similar to Fig. 5 and for identifying perilously dense or trapped areas.

In intervals a software controller changed the device's connectivity state to activate the p2p part of the protocol. That is, randomly after 40 to 60 s being connected, the device disconnected for 30 to 50 s and thereafter reconnected again. The controller worked only protocol-internally leaving the connectivity via the actual network interface unaffected. Only when the actual network connectivity changed, the control adjusted accordingly and dis/reconnected when the network interface did so.

4.1 Participation

With visitors coming and going, their participation in the experiment varies over time. Figure 4 depicts the participation behaviour as broken bars. A horizontal line of bars represents one device. Bars are broken when successive sensor readings are more than two minutes apart. The sensor readings, here, refer to the mock readings created by the phone. They reflect participation best because they are unaffected by the device's distance to real sensor sources and other peers. Some participants contribute right from the start, while others join-in later. Some hold out to the end, while others leave early or contribute only for a short period of time.

Participation gaps may occur because participants cancel their participation deliberately, e.g., out of curiosity to see the effect on app features. Or, participants pause participation involuntarily as they miss notifications on the renewal

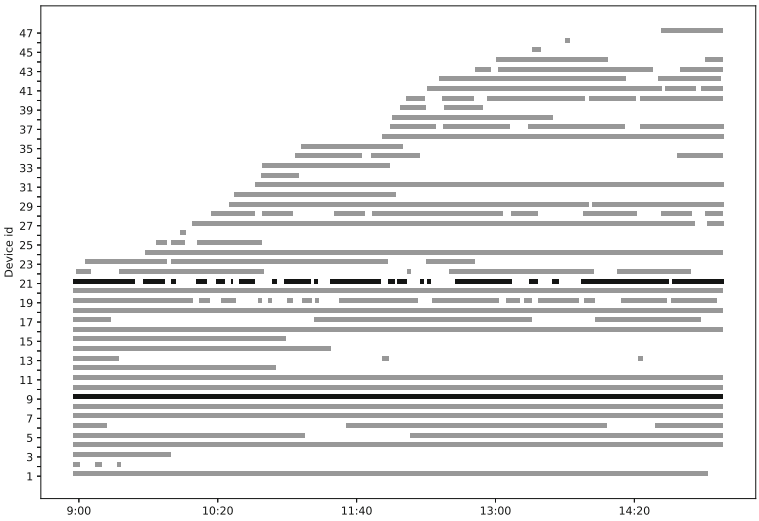


Fig. 4. Participation in the experiment varies: Some devices contribute throughout the event (e.g., device 9) while others do so only intermittently (e.g., device 21).

of app permissions. Expiring permissions, like for being Bluetooth discoverable, is among Android's efforts to protect their users' privacy. This, however, challenges apps with background features like a p2p protocol. They need to bring user attention back to the phone without negatively impacting the user's current experience in the real-world. Considering the staccato-like behaviour of the device 21 in Fig. 4, there may be other reasons, which are not yet clear.

4.2 Radio and Spatial Clusters

Over time participants formed clusters in terms of their spatial distance and radio range. Spatial clusters are derived from sightings of BLE beacons. A centroid technique [7] evaluates the beacons' received signal strength and positions each participant accordingly. Density-joining [10] clusters participants based on their neighbourhood within a two meter radius. For radio clusters, the peer sightings dataset is used, which defines neighbourhood as the number of peers a device senses in its radio range. Density-joining assigns those to the same cluster that have at least one peer in common.

Figure 5 depicts the number and size of the clusters over the course of the experiment. A dot represents a cluster. It grows in size and lightness, the more members the cluster has. In spatial terms, participants are rather scattered and independently roam the event area. This is reflected by the high number of small dots in the top part of the figure. The exception is past noon (12:20–12:50) when 10 to 15 participants gather in the same part of the event area. This may be due to a program item that catches the interest of many visitors. In contrast, the bottom of the figure shows the evolution of one large radio cluster and how

it grows to more than 20 members towards the end of the event at 14:40. There are only few small radio clusters with one to five members that are disconnected from the rest possibly because they are in the most remote places of the event area. The analysis of the radio clusters shows that most participants are part of the same p2p network, which allows for data exchange and task offloading.

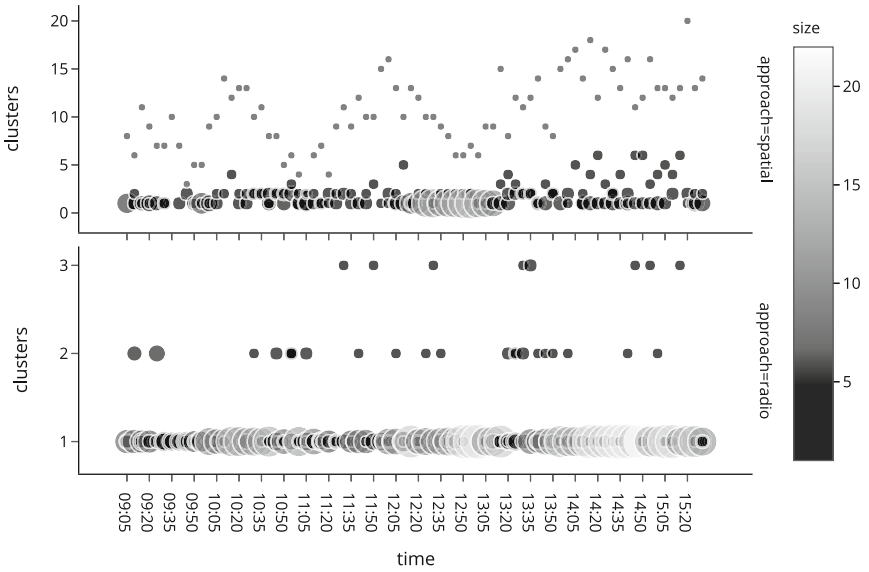


Fig. 5. Despite the participants' spatial distribution (top), most of them are part of one large radio cluster (bottom) leaving only few to small isolated radio clusters.

4.3 Connectivity Changes

With a computer science building as the venue for the event and experiment, the assumption is that network connectivity is not an issue. Disconnects that do occur, would not be sufficient to activate and sample the p2p part of the protocol in suitable quantities. A software controller thus induced artificial dis- and reconnects.

Analysing the connectivity state in the peer sightings dataset, Fig. 6 shows the median connectivity change rate over time. Considering the effect of the software controller, a rate of up to 1.5 changes per minute is expected. At times, however, the depicted values are much higher. This means, devices experience additional dis- and reconnects from the actual networking infrastructure. The peak times at 12:45 and 14:35 coincide with the times when the size of the radio and spatial clusters peak. This suggests that already at a medium size indoor event, network issues occur when the infrastructure is not particularly adjusted to the expected number of visitors. Typically, existing infrastructure is designed for the average use case that does not accommodate for crowds of people.

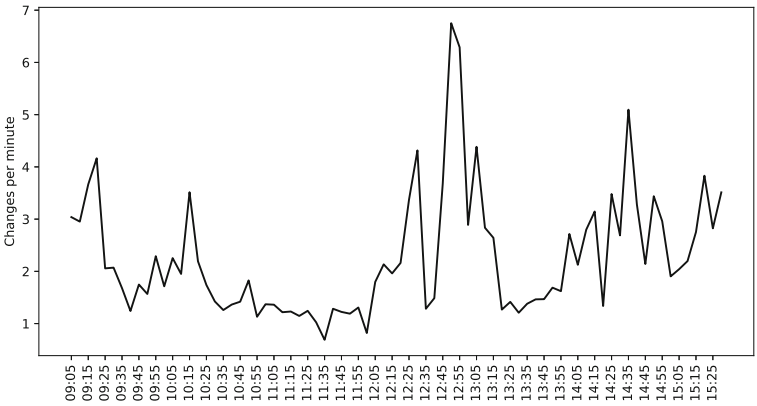


Fig. 6. The median connectivity change rate is at times much higher than the expected 1.5 changes per minute. This indicates that participants experience connection issues with the networking infrastructure.

4.4 P2P Transfers

Frequent dis- and reconnects trigger the protocol switch more often, which may increase the number of aborted p2p transfers. A p2p transfer is aborted in two cases: First, while the transfer is in progress, the remote end loses network connectivity and closes all its serving sockets to inform its clients about the change allowing them to find a new server peer. The client peer records this as an aborted transfer. Second, the device reconnects to the network, stops the p2p transfer, and uploads data itself rather than risking a disconnect from the remote end.

Figure 7 depicts the average count of completed and aborted transfers over time. The analysis focuses on the two most prominent cases which repeat themselves throughout but possibly to a lesser extent. It shows an increased number of aborts from 13:00 onwards, which means that more transfers are affected by the connectivity change. In contrast, the transfers at noon (11:25 to 13:00) are less affected since the number of completed transfers is higher than the aborted ones. One reason is that at noon some devices experience disconnects that are much longer than induced by the software controller. Whether they are in a blind spot or deliberately turned their networking interfaces off, is unclear. Apart from that, not every connectivity change happens during a p2p data transfer but instead to peers that are currently idle.

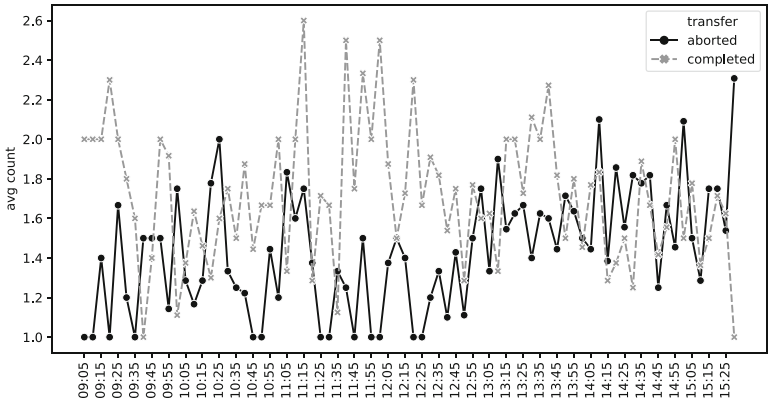
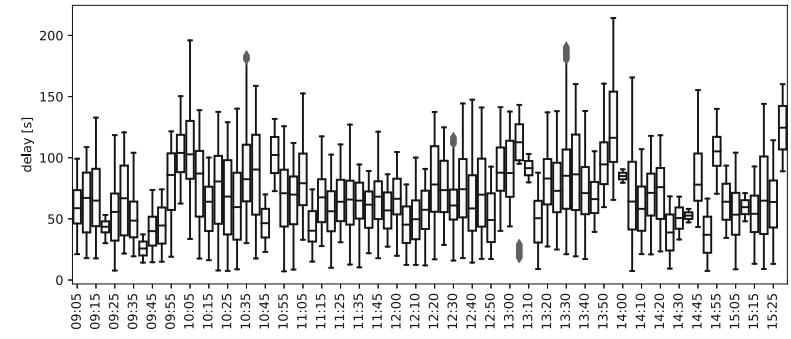


Fig. 7. At around noon, p2p transfers are much less affected by connectivity changes than from 13:00 onwards when the number of aborted transfers increases.

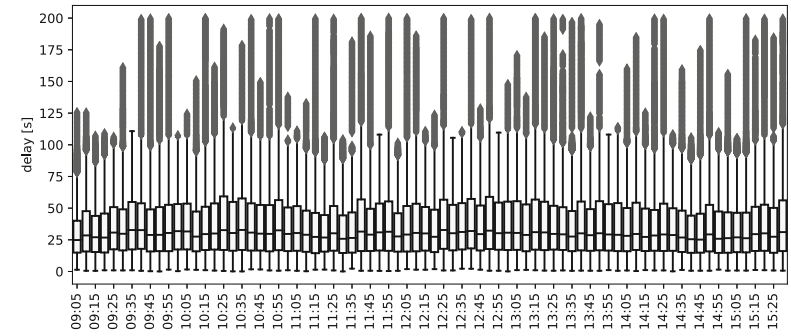
4.5 Data Delay

Data delay is the time from a senor reading being collected to when it becomes available at the server. The analysis refers to the mock sensor readings, which have been uploaded to the server either directly or via a peer. For completed transfers, the data delay is well below 200 s (cp. Fig. 8a). The mean delay is around 100 s. Given the small amounts of data transferred, this delay is rather high. This is caused by the timeout between protocol iterations. If a device did not discover a connected peer, it idled for 60 s before it run the p2p part of the protocol again.

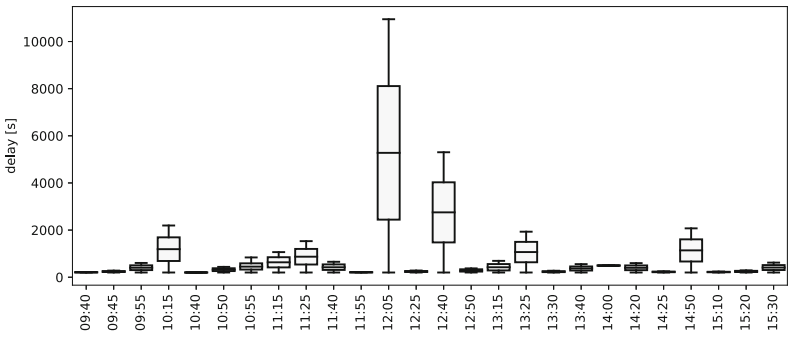
For data directly uploaded by the device itself, the mean delay is at 25 s, which is expected (cp. Fig. 8b). Notice, however, the density of outliers, which show that a considerable number of devices deviate from the average. One explanation may be that these devices started out with a p2p transfer, regained network connectivity, aborted the transfer, and uploaded data themselves. This takes longer than uploading data right away. There are, however outliers whose delay is too long to fit into the same graph (cp. Fig. 8c). In these cases, the devices remained disconnected for a long time, during which they were unable to find a connected peer in their proximity. Or, during which the payload had grown to a size that no peer connection was stable enough to complete the transfer, either because the peer moved out of range, or the peer lost connectivity. Generally, the phone creates 84 Bytes of mock readings per second adding up to 300 Kilobytes an hour.



(a)



(b)



(c)

Fig. 8. The delay for completed transfers and subsequent uploads would be rather swift if it was not for the timeout between protocol iterations (a). For direct uploads a considerable number of delays deviate from the otherwise short average delay (b). Some direct uploads take so long that they are outsourced to a separate graph (c).

5 Conclusion

The availability of crowd monitoring data is essential for event organisers and emergency response personnel to ensure safety. In case of network issues, data transfers to peers in vicinity that are still connected may allow for sharing such data nonetheless. This paper focuses on the delivery of crowd monitoring data. During an experiment at an indoor fair, a p2p transfer protocol was tested for how it meets real-world challenges. With considerable effort put into the design of the events' companion app, 47 visitors agreed to become visible to peers and share sensor and log data. The results of the experiment show:

- Despite variations in the continuity and duration of participation, most participants were part of the same large radio cluster throughout the experiment and thus fulfilled the main prerequisite for p2p data transfers.
- Even at this medium size indoor event, network issues occur when the infrastructure is not particularly adjusted to the expected number of visitors. This underlines the necessity for solutions that mitigate the loss of data.

Further, unanticipated changes in the connectivity to the network infrastructure challenged the protocol and highlight prospective refinements of its design:

- Participants experienced at times high rates of connectivity changes. The protocol quickly switched between dis- and reconnects often aborting p2p data transfers. This is inevitable when the remote end cancels the connection. However, losing the network connection could be approached more gracefully: Instead of immediately switching to a p2p transfer, a device could wait to see if it can reconnect after a short time. When the disconnect does take longer, a p2p transfer is feasible as the experiment shows short delays when a connected peer is around.
- Disconnects that lasted dozens of minutes caused the steady growth of data to be transferred. Agnostic of these changes, the protocol kept trying to transfer all collected data at once. For this to succeed, a peer connection would be required to be stable for an extended period of time. A more realistic way to mitigate this built-up of data is to split it up into small chunks that are suitable for short connections times with peers.

Overall, the lesson learnt from testing the protocol in a real-world setting is that attention to detail is important before advancing the protocol design e.g., to a multi-hop solution.

References

1. Blanke, U., Tröster, G., Franke, T., Lukowicz, P.: Capturing crowd dynamics at large scale events using participatory GPS-localization. In: IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing, Conference Proceedings (ISSNIP), pp. 21–24. IEEE (2014). https://doi.org/10.1109/ISSNIP.2014.6827652

2. Castagno, P., Mancuso, V., Sereno, M., Marsan, M.A.: 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), pp. 1–9, June 2017. https://doi.org/10.1109/WoWMoM.2017.7974296
3. Danielis, P., Kouyoumdjieva, S.T., Karlsson, G.: Urbancount: mobile crowd counting in urban environments. In: 2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 640–648, October 2017. https://doi.org/10.1109/IEMCON.2017.8117189
4. Groba, C., Springer, T.: Exploring data forwarding with bluetooth for participatory crowd monitoring. In: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 71–76, March 2019. https://doi.org/10.1109/PERCOMW.2019.8730711
5. Kannan, P.G., Venkatagiri, S.P., Chan, M.C., Ananda, A.L., Peh, L.S.: Low cost crowd counting using audio tones. In: Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems, SenSys 2012, pp. 155–168. Association for Computing Machinery, New York (2012). https://doi.org/10.1145/2426656.2426673
6. Kjægaard, M.B., Wirz, M., Roggen, D., Tröster, G.: Mobile sensing of pedestrian flocks in indoor environments using wifi signals. In: 2012 IEEE International Conference on Pervasive Computing and Communications, pp. 95–102, March 2012. https://doi.org/10.1109/PerCom.2012.6199854
7. Kluge, T., Groba, C., Springer, T.: Trilateration, fingerprinting, and centroid: taking indoor positioning with bluetooth LE to the wild. In: 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM) (WoWMoM 2020). Cork, Ireland, June 2020. Accepted
8. Loomba, R., de Frein, R., Jennings, B.: Selecting energy efficient cluster-head trajectories for collaborative mobile sensing. In: IEEE Global Communications Conference (GLOBECOM), pp. 1–7 (2015). https://doi.org/10.1109/GLOCOM.2015.7417727
9. Wirz, M., Franke, T., Roggen, D., Mitleton-Kelly, E., Lukowicz, P., Tröster, G.: Probing crowd density through smartphones in city-scale mass gatherings. EPJ Data Sci. **2**(1), 1–24 (2013). https://doi.org/10.1140/epjds17
10. Wirz, M., Schläpfer, P., Kjundefinedrgaard, M.B., Roggen, D., Feese, S., Tröster, G.: Towards an online detection of pedestrian flocks in urban canyons by smoothed spatio-temporal clustering of GPS trajectories. In: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks. LBSN 2011, p. 17–24. Association for Computing Machinery, New York (2011). https://doi.org/10.1145/2063212.2063220
11. Zhang, J., Guo, H., Liu, J.: Energy-aware task offloading for ultra-dense edge computing. In: 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 720–727 (2018). https://doi.org/10.1109/Cybermatics_2018.2018.00144