Franke *et al. Journal of Internet Services and Applications* (2015) 6:27
DOI 10.1186/s13174-015-0040-6

Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

Smart crowds in smart cities: real life, city scale deployments of a smartphone based participatory crowd management platform

Tobias Franke[1*], Paul Lukowicz[1] and Ulf Blanke[2]

Abstract

We describe a platform for smart, city-wide crowd management based on participatory mobile phone sensing and location/situation specific information delivery. The platform supports quick and flexible deployments of end-to-end applications for specific events or spaces that include four key functionalities: (1) Mobile phone based delivery of event/space specific information to the users, (2) participatory sensor data collection (from app users) and flexible analysis, (3) location and situation specific message multicast instructing people in different areas to act differently in case of an emergency and (4) post mortem event analysis. This paper describes the requirements that were derived through a series of test deployments, the system architecture, the implementation and the experiences made during real life, large scale deployments. Thus, until today it has been deployed at 14 events in three European countries (UK, Netherlands, Switzerland) and was used by well over 100,000 people.

Keywords: Crowd management, Crowd sourcing, Large scale cooperative sensing, Smartphones

1 Introduction

Pedestrian crowds are an integral part of cities. Planing for crowds, monitoring crowds and managing crowds, are fundamental tasks in city management. As a consequence, crowd management is a sprawling R&D area (see related work) that includes theoretical models, simulation tools, as well as various support systems. There has also been significant interest in using computer vision techniques to monitor crowds. However, overall, the topic of crowd management has been given only little attention within the smart city domain. In this paper we report on a platform for smart, city-wide crowd management based on a participatory mobile phone sensing platform. Originally, the apps based on this platform have been conceived as a technology validation tool for crowd based sensing within a basic research project. However, the initial deployments at the Notte Bianca Festival[1] in Malta and at the Lord Mayor's Show in London[2] generated so much interest within the civil protection community that it has gradually evolved into a full-blown participatory crowd management system and is now in the process of being commercialized through a startup company. Until today it has been deployed at 14 events in three European countries (UK, Netherlands, Switzerland) and used by well over 100,000 people.

1.1 Contributions

In this paper we describe the design, implementation and experiences with a large scale participatory platform for city wide crowd management. The platform enables the quick and flexible deployment of end-to-end applications for specific public spaces or events that include four key functionalities:

1. Mobile phone based delivery of event/space specific information to the users. In a nutshell, this is the classic city/event information app functionality that motivates users to download and run the App.
2. Participatory sensor data collection from app users (on an opt-in basis) and translation of that data into an appropriate representation of the crowd state and its evolution. This provides responsible crowd managers with a "heat map-like", concise, real time

*Correspondence: tobias.franke@dfki.de
[1] German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
Full list of author information is available at the end of the article

Springer

© 2015 Franke et al. **Open Access** This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

overview of the spatial and temporal evolution of the state of the crowd in the entire area.
3. Support for location and situation specific messaging allowing the messages to instruct people in different areas to act differently in case of an emergency. This allows for smart, adaptive emergency response and evacuation strategies to be implemented in real time.
4. The ability for post mortem analysis of an event as means of future planning.

On the research level, this paper makes the following specific contributions:

1. The concept of crowd management as a use case for participatory smart city technology including a requirements analysis based on several real life deployments of different prototypes.
2. The architecture and implementation of a corresponding platform comprising of
    - A generic, configurable, multi-platform app to facilitate collaborative, city-wide sensing as well as location and situation sensitive information delivery
    - A data processing backend that collects and stores the sensor data from the participants and converts them into situational awareness of the crowd state.
    - A configuration front end that allows the generic app platform to be quickly instantiated according to the specific event/space.
    - An event management front end that provides the visualization of the crowd state and allows for an easy creation of situation and location specific messages.

    All of the above is implemented in a modular way allowing easy integration of additional functionality (e.g. new sensor data analysis and visualization schemes).
3. Results and experiences from large scale real life deployments ranging from the information that the system can deliver about an event over the statistics of app usage to quantitative and qualitative results of user interviews (with both event visitors and civil protection personnel).

1.2 Related work

Obtaining knowledge about the current size and density of a crowd is one of the central aspects of crowd monitoring [1]. For the last decades, automatic crowd monitoring in urban areas has mainly been performed by means of image processing [2]. One use case for such video-based applications can be found in [3], where a CCTV camera-based system is presented that automatically alerts the staff of subway stations when the waiting platform is congested. However, one of the downsides of video-based crowd monitoring is the fact that video cameras tend to be considered as privacy invading. Therefore, [4] presents a privacy preserving approach to video-based crowd monitoring where crowd sizes are estimated without people models or object tracking.

With respect to the mitigation of catastrophes induced by panicking crowds (e.g. during an evacuation), city planners and architects increasingly rely on tools simulating crowd behaviors in order to optimize infrastructures. Murakami et al. [5] presents an agent based simulation for evacuation scenarios. Shendarkar et al. [6] presents a work that is also based on BSI (believe, desire, intent) agents – those agents however are trained in a virtual reality environment thereby giving greater flexibility to the modeling. Kluepfel et al. [7] on the other hand uses a cellular automaton model for the simulation of crowd movement and egress behavior.

With smartphones becoming everyday items, the concept of crowd sourcing information from users of mobile application has significantly gained traction [8]. Roitman et al. [9] presents a smart city system where the crowd can send eye witness reports thereby creating deeper insights for city officials. Szabo et al. [10] takes this approach one step further and employs the sensors built into smartphones for gathering data for city services such as live transit information. Ghose et al. [11] utilizes the same principle for gathering information on road conditions. Pan et al. [12] uses a combination of crowd sourcing and social media analysis for identifying traffic anomalies.

We have previously published papers focusing on the sensing part of our system. In [13] we describe in details how the data gathered using the mobile phones is being processed and visualized. In [14] the accuracy of our data processing and visualization method is evaluated using video footage from an event where the system was deployed. It is shown that while the user penetration of our system was below 1 % during this event our methodology nevertheless achieved a correlation factor of 0.83 between the actual crowd density (obtained by counting people on videos) and our system's calculated crowd density. While our previous work dealt with the matter of assessing crowd parameters in detail, this paper is concentrating on presenting the entire approach as a whole by providing insights into use cases, the overall system architecture and lessons learned.

2 The crowd management use case

Crowd phenomena are a well studied yet by far non-trivial example of collective human behaviors [15, 16]. Understanding and controlling such phenomena has significant practical relevance in civil protection applications such as, for example, the management of large scale public events

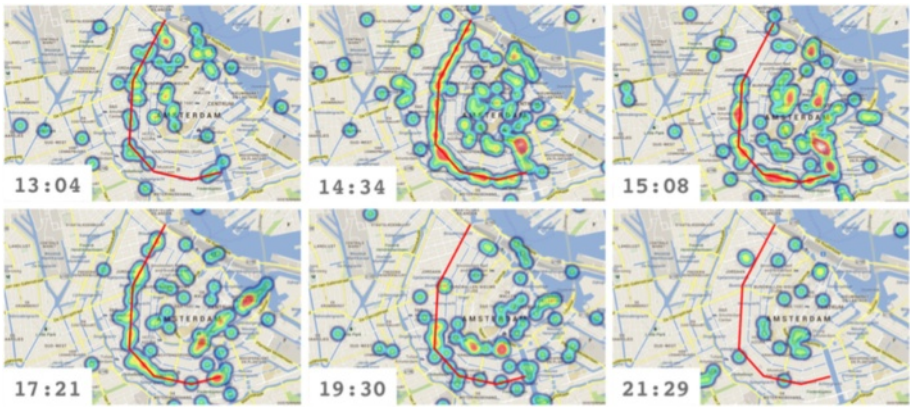

Fig. 1 Crowd state parameters visualized in form of a heat map. It can be clearly seen how the crowd state evolves over the duration of an event

and emergency evacuation. Problems with crowd management can have disastrous consequences. Well known examples of crowd disasters include the 1990 stampede in Mecca (where 1426 people died inside a pedestrian tunnel) [17] or the 2010 German Love Parade disaster where 21 people were trampled to death (with more than 500 having been injured) when a mass panic broke out [18].

Two key requirements for successful crowd management are situational awareness and the ability to exert influence on the crowd.

2.1 Situational awareness

In planning and managing crowds at large scale events, situational awareness goes far beyond the mere ability to observe the area in question [14, 19]:

1. Individual observations must be put together into a coherent picture that provides information about relevant global parameters such as density distribution, motion directions, turbulence etc.
2. Developments and trends need to be monitored and analyzed as they evolve over time so that problems cannot only be spotted when they occur, but be foreseen in time to be prevented.
3. It should be possible to review the entire course of an event retrospectively to identify problems and irregularities and understand their causes. While, fortunately, major accidents are rare, deviations from the plan, unexpected behaviors and potentially dangerous situations are common. Understanding how and why they happened is critical for reducing the probability of major incidents in the future.

Traditionally, situational awareness during public events has been based on observations and reports from civil protection forces deployed on site. Obviously, this lead to patchy, non-real time information with limited reliability as people sporadically report via radio what they see. As a consequence, CCTV cameras have nowadays become an indispensable tool, providing timely, reliable information that can be recorded and replayed later on. However, even with a large number of cameras, the information tends to be patchy (one rarely has complete coverage of the entire area) and most of all requires complex interpretation in terms of the overall coherent picture and global parameters. Thus, going from seeing dozens of parallel video streams from isolated locations to having a reliable global picture of the overall crowd density and motion is not trivial. Deriving trends, making predictions, or finding causes for problems in such streams is even more difficult. As a consequence, tools are needed that collect, aggregate and interpret information from the entire event area (and possibly beyond it) and automatically generate an easily understandable representation of the relevant global crowd state parameters and their evolution in time and space (see Fig. 1). One possible approach is the automatic evaluation of video streams to count people and even track their movement. Another one – pursued in our system – is to rely on (voluntarily provided) sensor information from peoples' smartphones.

2.2 Crowd control

The main tool of crowd management during large scale public events is the physical layout of the space defined and implemented in advance. It includes barriers, entrances, exits, gangways etc. In general, the layout is based on theoretical models and simulations [15]. Given (1) an initial crowd state (e.g. size, initial distribution), (2) a physical space layout, and (3) a certain general crowd behavior (e.g. everyone heading towards the nearest exit, or most people moving in one direction) such models can predict the evolution of global parameters such as the density distribution, evacuation time or average physical pressure within the crowd. During the event, crowd

management needs to react to changes in the crowd's state and behavior in order to make sure that key parameters remain within acceptable bounds and no undesirable phenomena (e.g., panic, stampede, blockages, jams, etc.) occur. To this end two main types of actions can be taken: (1) dynamic changes in space configuration (e.g. closing entrances, opening new exits, creating new barriers) and (2) issuing instructions to the crowd in order to influence its behavior. While, within certain limits given by the venue, physical space reconfiguration can be prepared beforehand (e.g. placing removable and movable barriers, or posting security personnel to redirect people) and executed effectively, issuing complex instructions to the crowd and ensuring compliance is a difficult problem. Today, it often involves "primitive" approaches such as security forces shouting through megaphones and raising improvised signposts. Obviously, the amount and complexity of information that can be delivered this way is limited. In particular it is difficult to deliver differentiated instructions (e.g. send different parts of the crowd to different exits) and background information explaining the necessity of certain measures (crucial to ensure compliance). By contrast, a location and activity sensitive App can deliver complex, personalized and situation adapted instructions and convincing explanations in real time.

3 Participatory app based crowd management system

Our work is based on the observation that more and more cities, tourist resorts, sports clubs, concerts and parades have their own apps and people increasingly rely on such apps to plan and manage their visit. Many such apps already use sensors (in particular location) for context and situation adapted information delivery. As a consequence they are a potentially perfect vehicle for both collecting data about a crowd and delivering personalized, situation specific information and instructions. A key research question addressed in this paper is what is needed to realize this potential. The answer is based on a series of deployments during which our system has been incrementally evolved and validated building on feedback from various stakeholders (for details, see the box "How was the System perceived?") and usage by around 100,000 people until now [20, 21].

3.1 Basic considerations

The core functionality of the proposed system is the estimation of crowd density distribution from individual location fixes provided by a subset of users (the ones who have installed the app). From the point of view of practicality the two main questions are:

1. How many users need to install the app to provide a useful estimate?
2. How much communication load does the system generate?

3.1.1 Required number of users

In simple terms density estimation means that, for any given area within the relevant space, we can provide the percentage of visitors who are located there. Our system determines this percentage from the sample of users who are running our app: if x percent of the users running our app are within a given area, we assume the same to be true for all users. Thus, we essentially perform statistical sampling which is a common technique for example in opinion surveys and for which the relationship between sample size and the accuracy of the estimate is well understood.

For a sufficiently large sample size n, the distribution of a population proportion will be closely approximated by a normal distribution and therefore the margin of error for the estimation of a population proportion (ME_p) can be calculated with the Wald method using the formula

$$ME_p = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \cdot z_{\alpha/2}$$

where $\hat{p}$ is the sample proportion and $z_{\alpha/2}$ is the 100 $(1-\alpha/2)$th percentile of the standard normal distribution (the confidence level). Since we are aiming for a confidence level of 95 % and we do not know the actual sample proportion, we use $z_{\alpha/2} = 1.96$ and set $\hat{p}$ to the worst-case percentage 50 %. Hence, the formula becomes

$$ME_p = \sqrt{\frac{0.25}{n}} \cdot 1.96$$

Thus, if the ME_p computed from the above formula is for example 1 % then, with the probability of 0.95 (our chosen confidence), the true number of people within a given area will be within ± 1 % of the percentage estimated from the data provided by the app users.

Looking at the deployment during the 2013 Zurich festival as an example, a total of 28,000 people were uploading location data into our system – with about 4,000 simultaneous uploaders at peak times (see Fig. 11 towards the end of this paper for reference). Setting n = 4,000 leads to ME_p = 0.0155 – an error margin of roughly 1.5 %. The total number of visitors over the three day festival period was roughly 2 million, which equals about 660,000 unique visitors per day with a peak of approximately 300,000 people being at the festival area at the same time. Hence, at a confidence level of 95 % our system has an error margin of roughly 0.0155 × 300,000 = 4650 people during peak times.

To understand the significance of this number we need to look at the type of density effects that are relevant for event planning and crowd management. According to [22], large scale events are planning with an average crowd

density of 2 persons per square meter. In some countries (e.g. Germany) this value is even part of the law. While 4 persons per square meter are considered as very crowded, [22] furthermore states that in reality the crowd density hardly ever surpasses 6 persons per square meter (which is considered as critically crowded). In order to provide a feeling for those density values, Fig. 2 shows two examples of crowd densities recorded during public viewing events.

Summarizing the above, for crowd management purposes we need to be able to reliably distinguish between a number of different density states (e.g. less than 2, around 2, around 4, around 6 persons per square meter, etc.). How does that relate to the margin of error described above? The answer is: through spatial resolution. The margin of error is given as absolute number of people. Integrating the density over an area also gives us an absolute number of people. The larger the area the larger not only the absolute number of people but also the absolute number of people by which the relevant density states differ.

Coming back to the specific example from the Zurich festival, we consider a zone with an area of 75 m by 75 m. This leads to 11,250 people per zone in case of a density of 2 persons per square meter. The worst-case number of 6 persons per square meter leads to 33,750 people per zone. With our margin of error we can estimate the number of people within a given area to within ± 4650 people. This allows us to easily distinguish the relevant states, as the distance between them is more than double the margin of error.

In the following we want to give a better general feeling for the parameters. Table 1 shows how different zone sizes influence the number of people within each cell and consequently the minimal sample size that's needed in order to detect critical states with the same quality as it was achieved in the Zurich example. In summary, the more people are in a cell, the more levels of density our system can distinguish. Of course the trade-off is that this higher resolution in crowd density comes with a loss of spatial resolution. Figure 3 demonstrates the influence of the sample size on ME_p at a fixed confidence level of 95 %.

Table 1 Influence of zone size on the number of people within a zone and the minimal sample size min_n which is needed to detect critical states

	Persons per m^2			min_n
	2	4	6	
25 m x 25 m	1250	2500	3750	320,000
50 m x 50 m	5000	10,000	15,000	20,000
75 m x 75 m	11,250	22,500	33,750	4000
100 m x 100 m	20,000	40,000	60,000	1260
125 m x 125 m	31,250	62,500	93,750	520

3.1.2 Communication load

The second point we want to discuss with respect to basic considerations is our approach's data consumption. If the system uses too much of the user's data plan, it will not be accepted on a wide basis. As it will be shown later on in this paper, there are two main data streams which need to be analyzed for this: (1) location data being uploaded to our server and (2) content updates (i.e. event specific information such as schedules, maps, etc.) and messages which are sent from the event organizers to the app users.

With respect to content updates and messages, making a general statement is impossible as that amount of data very much depends on the specific event. Usually, apps are delivered with all the contents already on board, hence

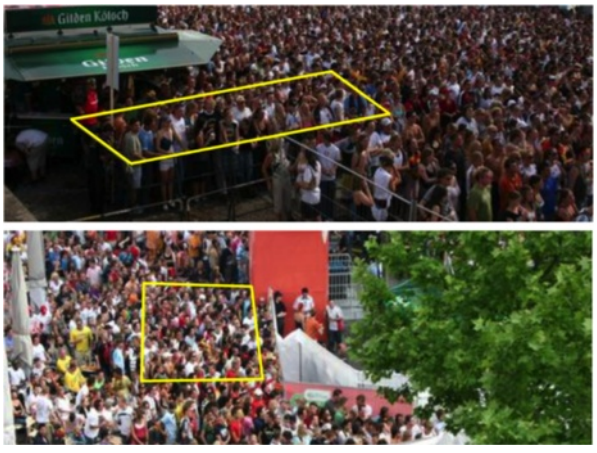

Fig. 2 Examples of different crowd densities during public viewing events. *Top*: 3.8 persons per m^2, *bottom*: 5.0 persons per m^2

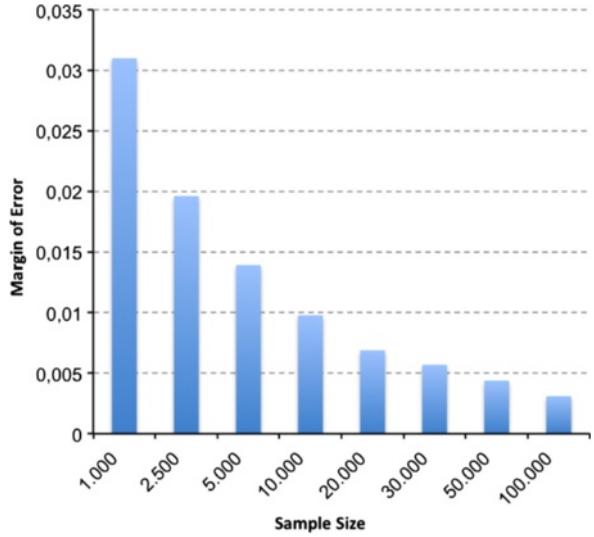

Fig. 3 Influence of sample size on ME_p

the amount of additional data traffic required for content updates depends largely on how much information needs to be added to the app after its publication to an app store. The same is true for messaging: it is impossible to generalize the messaging behavior of an event organizer. However, given that messages and content updates are consisting of 90 % textual information, the data volume is definitely not exuberant. However, since the apps poll the server backend for content updates and messages, it is possible to quantify the amount of data that is being used for those update requests.

Table 2 summarizes the amount of data traffic caused by our system. Please note that those numbers include both the amount of data consumed for sending the request as well as for receiving the server reply. It can be seen that even in cases where the system is active for 24 h, the resulting traffic is very low – especially considering the fact that most smartphone users have flat rate contracts nowadays.

3.2 Evolution and deployment history

As described in the introduction, the system was originally developed as a participatory data collection test platform. As such, it was first deployed during the Notte Bianca festival in Malta. In a simple festival information application it embedded the ability to collect (with user informed consent) time stamped location data and transmit it to a remote data server. In addition, CCTV cameras were set up at key locations to collect crowd density ground truth.

After the event, a simple visualization of the crowd density from the smartphone data was compared with the video footage (see Fig. 4) and a survey was conducted to asses how users perceived the system. Both were discussed with the civil protection forces in charge of the festival. Two main conclusions have emerged from the above:

- Despite a very low penetration of the software (only less than one percent of the visitors had used it), at least at a qualitative level, the estimate of the crowd density was surprisingly accurate.
- The system was very well received by the civil protection forces (Civil Protection Department of Malta and the Ministry of Home Affairs). They particularly emphasized the importance of the real time crowd density information for identifying areas where incidents might be most likely to occur.

Table 2 Data traffic consumed by app-based crowd management system

	Upload	Update
Traffic per call	1.0 KB	1.1 KB
Traffic per hour	60.0 KB	66.0 KB
Traffic per day	1.4 MB	1.6 MB

From the above conclusions it was decided to move from a mere data collection platform to a full blown crowd and event management application. Therefore, a presentation was given to decision makers responsible for public safety in the City of London. Its goal was to find the right event that would (1) be willing to support our team during the creation of the application with feedback from an organizer's point of view and (2) have the ability to deploy and evaluate the application at a large scale.

The Lord Mayor's Show was identified as an ideal candidate. During a series of workshops with the event's organizers, the following main requirements were elaborated for the event management system:

1. It should not only consider emergency aspects but instead cover the whole bandwidth of event information. This also included the application's ability to adapt to last minute changes which are quite common in event management.
2. It should facilitate a unidirectional communication channel from the event organizers to the event visitors to be able to control the crowd efficiently by sending commands to the visitors' smartphones. Special emphasis was given to the need for location based messages which allow for sending a message only to those people located in a specific geographic area.

Given that during the time of the workshops several new opportunities for deployments arose (e.g. at the West End Live Festival in London or within the Westminster 2012 Olympics app), it became quickly apparent that creating a new standalone app for each deployment would not be feasible. Consequently, the decision was made to add another requirement for the event management system:

3. It should be based on a generic framework which could be quickly adapted to the requirements of each event without the need for any programming whatsoever.

Since the smartphone app part of the system should be usable by as many people as possible, it would have to run on the majority of smartphones. In practice, this means that the app needed to support the Android and the iOS operating systems which currently make up for roughly 96 % of the smartphone market. As the main scientific goal of the application was gathering sensor data, the smartphone apps needed to be implemented natively since the hybrid development approach has too many disadvantages with respect to accessing platform specific hardware

Fig. 4 Crowd density visualized compared with video footage recorded at the Notte Bianca festival 2011 in Valetta, Malta

features. Consequently, a fourth and last requirement was added to the list:

4. The framework should support the creation of native smartphone apps for the iOS and Android operating systems.

Based on the findings of the workshops and the list of requirements mentioned above, the system presented in this article was created. It mainly consists of two components. Firstly, a web application focusing on the needs of event organizers and emergency and civil protection staff offers the means to (1) define the design and the functionality of the smartphone app to be deployed and (2) to analyze the collected data from the crowd and to interact with the crowd via messages. For details about the web application's functionality, please refer to the sections 4.2 and 4.3.

The second main component of the system is the generic app for the iOS and Android operating system which is deployed at the events. Based on the outcome of the workshops, a set of modules for the app was defined (see section 4.1 for a complete list). At runtime, the generic app receives information about its design, menu structure and activated modules enabling it to present itself to the user in exactly the way intended by the event organizers. This ability of the app to reconfigure itself at runtime allows event organizers to change contents of the app at any time – even when the event is already taking place can information or features be added remotely to react to the current circumstances.

During the first deployments, the user base of the system was quite small (a couple of thousand app users per event). However, the deployment during the inauguration festivities in Amsterdam for King Willem of the Netherlands in April 2013 was the big breakthrough: thanks to an integration of the app into the event's publicity campaign a total of more than 70,000 people downloaded the app (about 10 % of the event spectators). Regardless of these numbers we were unable to actually perform any data recordings during the event because the largest Dutch telecommunication provider opposed to the system due to fears of network congestion on the highest political level. While these fears were cleared eventually, it was too late to perform a data recording.

Therefore, the first big impact deployment of the system was during the 2013 Zurich festival with a total of 28,000 active app users contributing data over a period of three days [21]. The resulting data set is the largest of its kind according to our knowledge.

Since the Zurich deployment, the system has been routinely used in several European events and the interest in the technology is rapidly growing. Furthermore, a Windows Phone app has been added to the framework.

Table 3 gives an overview about the deployments that have been carried out until the end of 2014. Please note that a sizable part of the deployments were labelled as "minor deployments" where the research team had no control about app distribution and merely provided the technology for interested partners. As a consequence of this, no solid numbers with respect to number of visitors and number of app downloads are available since this information was not disclosed. Only the "major deployments" were under full control of the researchers. Also, the number of visitors for each event (where available) are estimates as all of them were non-ticketed events without access control – hence, no overall ground truth data was available.

4 Basic functionality

Summarizing the above, the system's functionality can be divided into three categories: (1) features for the event visitors, (2) features directed at the event organizers and

Table 3 Overview of the system's deployments between 2011 and 2014

Event name	# Visitors	# App downloads	# Data contributors
	Major Deployments		
Notte Bianca, 2011	≈ 100,000	≈ 1000	≈ 340
Lord Mayor's Show, 2011	≈ 500,000	≈ 3000	≈ 830
Lord Mayor's Show, 2012	≈ 500,000	≈ 1000	≈ 920
Zurich New Year's Eve, 2012	n.a.	≈ 5000	≈ 3000
Dutch Coronation, 2013	≈ 750,000	≈ 70,000	n.a.
Zurich Festival, 2013	≈ 2,000,000	≈ 56,000	≈ 28,000
	Minor Deployments		
West End Live Festival, 2012	n.a.	n.a.	≈ 950
Westminster Olympics App, 2012	n.a.	n.a.	≈ 2300
Vier Daagse Feesten, 2013	n.a.	n.a.	≈ 2100
Amsterdam Gay Pride, 2013	n.a.	n.a.	≈ 1800
Tilburgse Kermis, 2013	n.a.	n.a.	≈ 1500
Koningsdag, 2014	n.a.	n.a.	≈ 4600
Vier Daagse Feesten, 2014	n.a.	n.a.	≈ 2200
Leids Ontzet, 2014	n.a.	n.a.	≈ 1700

(3) features for the emergency and civil protection services.

4.1 Visitor features

Visitors of an event are exclusively using the app element of the system. From their point of view, the app provides the services they need in order to get all the necessary information about the event. The actual feature set of the app differs between events and also depends on the event's requirements (see Fig. 5 for some examples). The following list contains all features which can be included in apps using our framework:

- The Map Module displays geographic contents such as points of interest (POIs) and routes either on a platform specific map (i.e. Google Maps, Apple Maps or Bing Maps) or on a custom map provided by the event organizers. POIs can be annotated with extra information such as images, texts and links. Furthermore, the user can navigate to POIs.
- The Event Calendar Module gives users an overview about the details of events that stretch out over a longer period of time. In case of the app for the London 2012 Olympics for example, it contained all competitions, concerts and other relevant events with detailed information such as maps, images, texts and links.
- The One Day Event Module provides a list based overview about the proceedings of short events.
- The Contact Module allows users to contact the event organizers and other important entities via phone or email and also gives them the option to visit their website.
- The Checklist Module can be used to inform visitors about items they need to bring along. The City of London Police for example uses this module in their app to tell people about the items they should have in their houses in case of emergencies.
- The Messaging Module acts as a generic inbox for broadcast and location based messages sent by the event organizers and the emergency forces. Messages

Fig. 5 Screenshots of selected app features from the Westminster 2012 Olympics app (from left to right: London travel planner, map, event information)

are being received via several channels (travel, general, emergency, location specific) to allow for an easy prioritization by the users.

- The RSS Reader allows app users to read event-relevant news that are being published as RSS feeds.
- In some cases, event organizers want to make Static Contents available for the visitors via the app. One example for this might be the map of a city's subway system.
- The City of London Police had a special module made for their app: the Street Level Crime Viewer. This module allows app users based in the UK to view the crimes that took place in an area of one kilometer around their current position allowing them for example to check if an area is known for frequent burglaries before moving there.
- Given the fact that most app users are very enthusiastic about social media, a Twitter Module was added to our framework to give users access to news spread this way.
- One of the easiest ways to add existing contents to our apps is by using the Web Content Module. The app for the London 2012 Olympics for example, incorporated the city's public transport information system using this module.
- A special case of the Web Content Module is the Mobile Heat Map Module which displays the crowd density heat map with a fixed view port.
- The Badge Collector is a great module to motivate event visitors to use the app more frequently. It sets certain goals for the users (e.g. spending a certain amount of time in a certain region) and awards them with badges if they reached that goal.
- The Friend Finder is another motivational module: it allows to select some of the user's Facebook friends and share the user's location with them. This enables app users to coordinate their stay at the event better with their friends.
- The app framework's Privacy Modules have the purpose of informing the app users about what exactly the app is doing. They give a detailed overview about the status of each sensor, show information on when and where a recording is being performed and also give the option to opt out of the crowd sensing feature.

4.2 Event organizer features

Event organizers mostly interact with the web application element of our system. Before an event, their main task is to create apps and fill them with content. To learn more about this process, please refer to the box "How to Manage a King's Coronation?".

During an event, our system gives event organizers the option to change their app's behavior and content with the click of a button. This way, organizers can add for example geographical contents to the app's map section to mark certain locations as closed or to highlight routes people should take to move to different parts of the venue.

Furthermore, event organizers need to be able to communicate with the visitors directly via messages. This task is also accomplished through the web UI. Messages can either be sent as broadcasts (please also see the box "How to Manage a King's Coronation" for details) in which case they are received by all app users or as location based messages, meaning that they are only received by people who are located within the geographic area targeted by the message.

4.3 Emergency and civil protection features

Like event organizers, emergency forces mostly use the system's Web UI component for their work during an event. They too can change the behavior of the app – however, the reason for doing so is much more serious.

Imagine an actual incident where important safety advice needs to be sent out to event visitors as quickly as possible. Using the Web UI, the emergency responders can disable all app features that are not directly relevant to the mitigation of the potentially hazardous situation, thereby focusing the users' attention on the important information.

The messaging features – especially the location based messages – can be used to steer different parts of the crowds into different directions to decrease the crowd density in critical locations. Adding geographic contents such as escape routes and safety zones to the app's map module is another valuable feature for crowd managers during an incident.

The most frequently used feature from an emergency and civil protection officer's point of view however, is the crowd density heat map. It enables those officers to gain an instant overview about the crowd behavior through an easy to understand visualization. This feature allows crowd managers to deploy their personnel on the ground in an efficient and sensible fashion as they can react to the crowd's movements. Figure 6 shows an example of such a heat map.

After an event is over, our system can play back the recorded crowd data allowing for a "post-mortem" analysis of the event. During numerous deployments, this feature proofed itself to be a unique tool to learn from an event and to improve crowd management related measures for future events.

How to Manage a King‘s Coronation?

The app framework allows for a fast and easy creation and administration of cutting-edge smartphone apps. In the following, we’re demonstrating how the official app for the coronation was built for and used during the event.

1 In your coding IDE, open the framework’s project file and define the app’s dispatch ID in the code. Compile the app and install it on your (and your client’s) test devices.

2 Log into the app framework’s cloud-based Web UI and create the new app by entering a name and the dispatch ID you used in the previous step. The app is now connected to the web UI.

3 Define the app's look and feel in the App Builder (see Fig. 1) by creating tabs and assigning features to each one of them. Continue by setting background images for menus, by changing the UI's color and by setting an icon for each tab. Click on the "Publish" button when you're done. Watch the app on your test device adopt the new look and feel almost instantly.

4 Set the contents for each feature you chose in the previous step. For example, in case of the map feature, you need to use the Map Editor (see Fig. 2) to place points of interest and assign an icon, a description, a picture and a link to each one of them. Click on the "Publish" button when you're done. Do this for all remaining features.

5 When you're happy with the initial design of the app, publish it to the app stores you want to support (Apple App Store, Google Play Store, Windows Phone Marketplace). Your app is now ready to be downloaded by the users!

6 In case of last minute changes before the event, you can adjust all the settings and contents via the Web UI – they will be pushed to the app users without the need for an app update upon the next start of the app. Change anything you want anytime you want: go wild!

7 If you want to schedule a recording session, access the Web UI's "Activity" menu (see Fig. 4) where you can draw a polygon onto a Google map to define the geographic boundaries of your recording session. Below the map you can enter the time frame where you want the recording session to be active. Click the "Publish" button and all the apps will receive the information upon their next start.

8 During a recording session, you can observe the crowd density heat map by clicking on the respective button in the Web UI's main menu.

9 If you want to send a message to visitors of the event, simply access the "Messaging" menu (see Fig. 3) in the Web UI and click on the "New Message" button. Enter the message text, an optional image and a link into the message composing window. If the message needs to be sent via push notification, make sure to tick the respective checkbox before clicking the "Save Message" button.

Fig. 1 – The App Builder


Fig. 2 – The Map Editor

Fig. 3 – The Messaging Menu

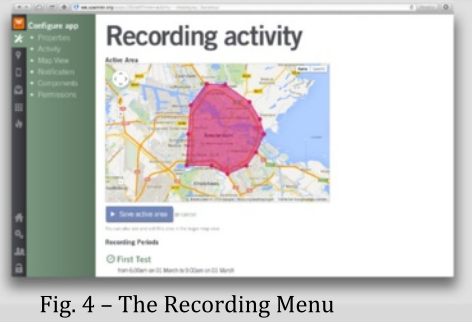

Fig. 4 – The Recording Menu

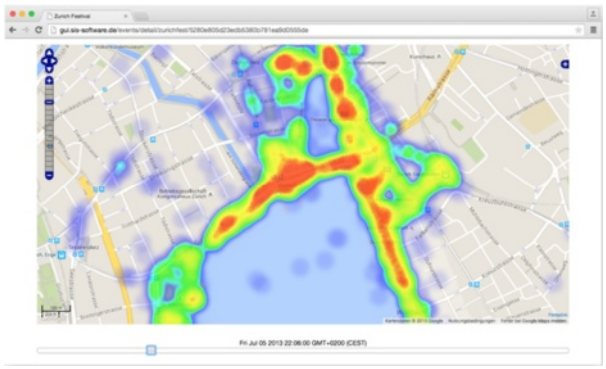

Fig. 6 Sample heat map

5 System architecture and implementation

In the following, we are describing the architecture of the system's three main components: the generic app which can be easily customized to the events' specific requirements, the data processing back-end and the web front-end which is used by event organizers to access the system's features. In contrast to existing crowd sourcing solutions (see related work section for examples) which are tailor made for one specific use case, our system architecture allows for a quick adaptation to the requirements of the user. This allows for flexible deployments in all sorts of events and smart city scenarios (where crowd control and targeted messaging are also of high value) without the need to re-engineer either the app component or the back-end. The entire process of creating an app (for details, please refer to the box "How to Manage a King's Coronation?") only takes a couple of hours at most with the biggest effort being the entering of event specific data (schedules, geographic information, etc.) into the system via the web front-end. Details about the system's unique flexibility are mentioned in the following sections.

5.1 Generic app

The generic app is the foundation of all event apps generated with our framework (see Fig. 7). It consists of four major parts: (1) the central app manager, (2) a sensor manager dealing with all tasks related to sensing and uploading data, (3) a statistics manager responsible for gathering usage data and sending it to the system's cloud based back-end and (4) layout and update managers responsible for dynamically reconfiguring both the apps' look & feel and their contents.

The biggest challenge was to make the generic app as dynamically reconfigurable as possible in order to quickly deploy event apps. At the same time the entire configuration process must be executable by people without programming skills. We reached that goal by encapsulating all the information about the app's layout, used modules and contents for each module in JavaScript Object Notation (JSON). The JSON files are generated through the web application element of the system (see section 5.3) in a user friendly fashion.

The updating mechanism of the app is realized by its update manager which controls a whole host of so-called update channels. Each update channel corresponds to one modality of the app – e.g. layout, map content, technical information, etc. The update channels poll the cloud based backend in regular intervals to check if the information the app has stored is still up to date. In case of an update, the new JSON file is passed to the update manager which stores it in the central data store object and notifies all relevant objects of the new data. These objects are then responsible for (1) retrieving the new data and (2) for reconfiguring themselves accordingly.

The layout manager is responsible for setting the app up according to its layout JSON file and is one of the first objects to be created during the app's startup procedure. It parses the JSON to get the information about the basic app structure – i.e. colors of UI elements, icons and names of tabs, menus, menu items and background pictures. When a user selects a menu item, the layout manager creates the corresponding object in an "get instance by name style". Consequently, the different event apps are always compiled with all modules "on board" – only at runtime it is decided which objects are actually needed and which ones aren't.

The app modules all inherit from a MainView object which encapsulates common functionalities such as notifying the statistics manager when a feature has been invoked. Furthermore, all app modules implement an interface defining the proper initialization routines. During its initialization, an app module reads the startup information delivered with the layout JSON to present itself correctly. For example in case of the web content module, this startup information consists of nothing more than a URL that should be displayed by the module.

The sensor manager is not only responsible for recording data from different sensor modalities and for forwarding this data to the upload manager. It is also responsible for deciding when and where the sensors must be activated and deactivated – in light of recent privacy discussions this is an especially important task. Based on the recording schedule JSON, the sensor manager enables the phone's rough location sensing modalities (i.e. energy aware location sensing through WiFi and GSM triangulation). Once the user is near the recording zone, the sensor manager switches to precise location sensing to get a clear idea about whether or not the user is in that zone. If the user is in the recording zone and has also consented to sharing his phone's sensor data, the sensor manager enables the recording of data. This whole process is also time triggered meaning that if the user is in a recording

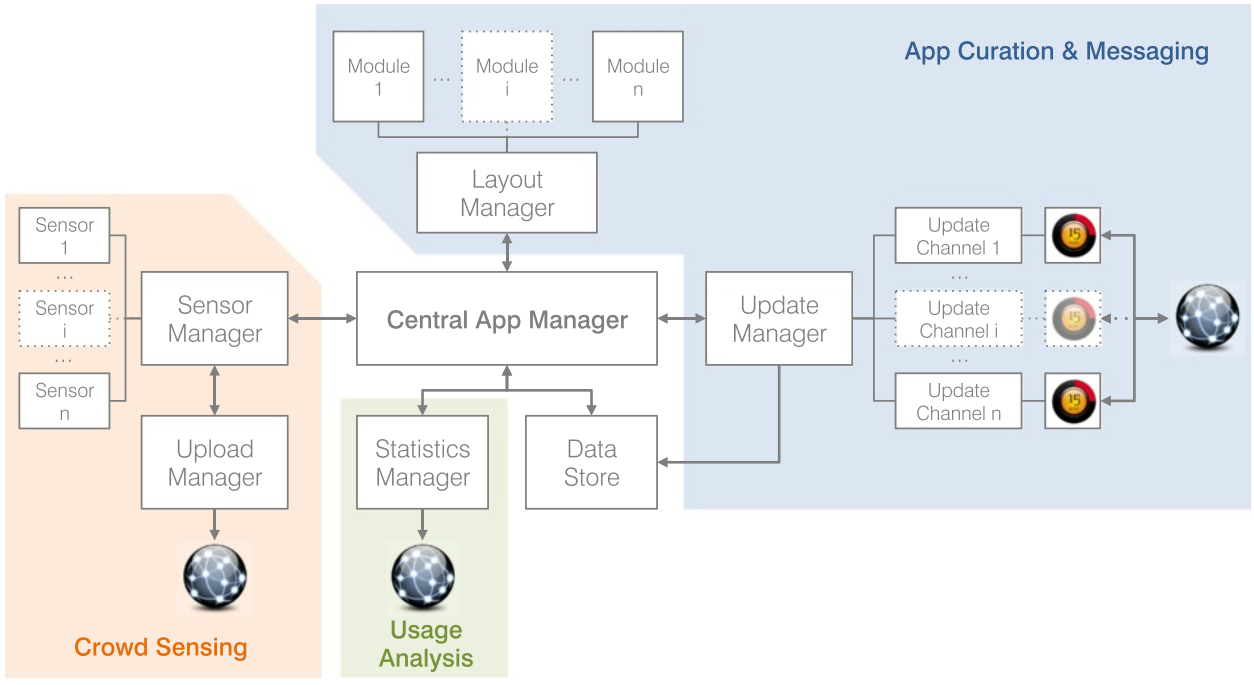

Fig. 7 Architecture of the generic app

zone but there's no recording scheduled at the current time, the sensor manager won't activate the data recording. Furthermore, the entire checking process happens on the device. In other words, this means that the devices are responsible for checking their geographic location and the current time and for comparing those parameters to the recording schedule. Consequently, no data is leaving the phone at any time until the user agrees to participating in the recording.

During a recording the data is stored in an upload buffer by the upload manager. This buffer is sent to the server once every minute. At the moment, the recorded sensor data is platform specific. On iOS and Windows Phone, our system only records GPS data including the user's heading, speed and the GPS fix's accuracy. On Android however, the app also offers the option to record the number of Bluetooth devices in the phone's vicinity. This data offers some valuable insights into the crowd density in the user's direct proximity.

The statistics manager has the simple job of keeping track of when and where each feature of the app has been used. Each module sends a corresponding notification to the manager. If a user stays on a screen for more than three seconds, the manager considers the feature as having been used and creates a record for it. This record contains the name of the feature, the time of use and the location of the usage. In case of navigation and POI information requests, the record also contains the location of the navigation's target or the POI's name and location. This information can be used to gain greater insights into how the app was used and allows for a more detailed post-event analysis. Once every three minutes, the statistics manager sends the collected data to the system's backend. For the sake of privacy, the user ID is not being sent along with the data so it's impossible to create a precise usage record of individual people.

Finally, the central app manager is responsible for controlling the app's run loop. It registers with the smartphone platform's push notification service, initializes the main objects and controls the app's transition between background and foreground behavior.

5.2 Data processing backend

The system's backend needs to be considered as two separate sub-systems: (1) the system for administrating the smartphone apps (i.e. for creating the layout and content JSON files) and (2) the system for collecting and analyzing the recorded sensor data whose output is, for example, the crowd density heat map.

The administration system consists of an application server, an SQL database and a dispatching cluster. In all deployments up to now, we used Amazon S3 as storage system for binary files to minimize the load on the system's backend.

The application server runs the web application which will be described in the next section. The outputs of the web application are the app configurations which are being stored in the SQL database.

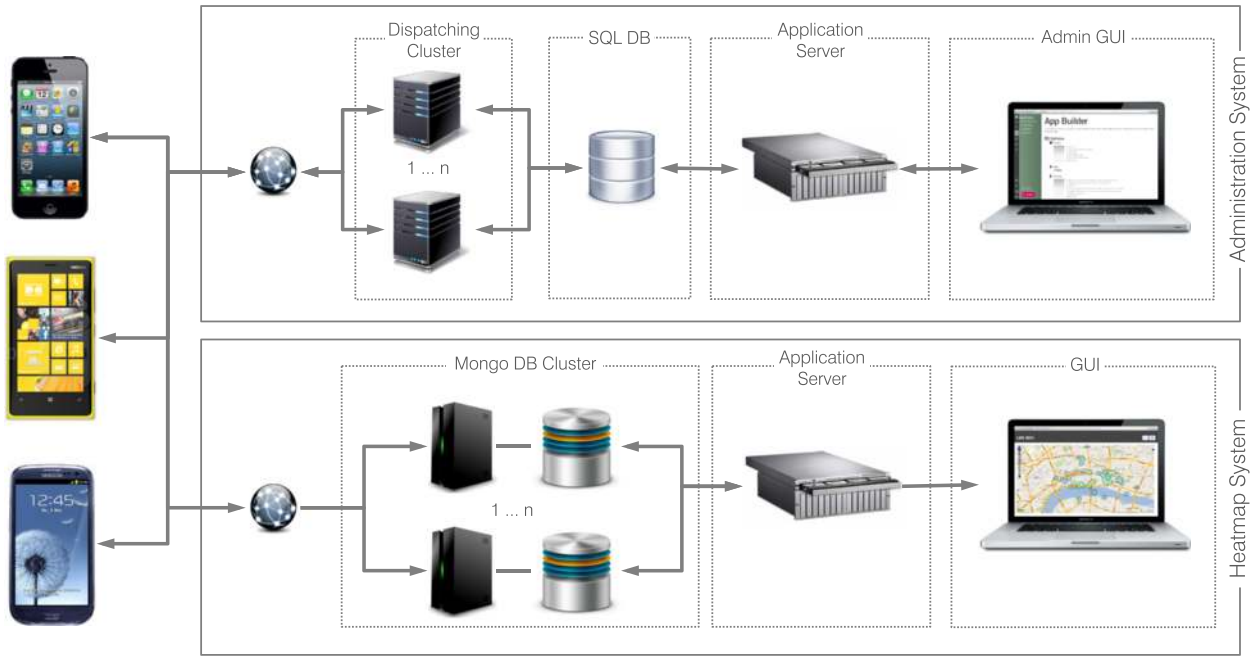

Fig. 8 Overview of the backend architecture

The dispatching cluster is the component with the heaviest load in this sub-system as all deployed event apps contact the cluster in order to check for updates. Versioning of app configuration data has been implemented using revision numbers – the app's update channels send a request to the dispatching cluster containing the app's dispatch ID and the revision number of the update channel. The dispatching cluster then compares the received revision number with the content's current revision number and sends back the new data in case of an update. Otherwise, it sends back an empty reply, thereby signaling to the client that the data is still up to date.

In order to perform as strongly as possible, the dispatch nodes try to keep the app contents in memory at all times. Whenever there's a change in the SQL database, the dispatch nodes get a notification letting them know that there is new data to dispatch. The nodes then create new JSONs for each affected update channel and keep that data in memory. This way, each update request can be handled by simply comparing two integers and potentially sending back pre-computed data.

The entire app administration sub-system is being hosted on Amazon's AWS cloud services for reasons of cost effectiveness and to make sure that there are always enough computing resources available to be able to cope with peaks in user load. Following the advise of Amazon consultants we decided to employ a multitude of smaller computing instances instead of a small number of large instances. When using small Amazon EC2 instances (roughly comparable to single core 1.5 GHz CPU machines with 1.7 GB of RAM) each instance is able to handle approximately 800 users. The dispatch cluster scales automatically based on the average load on all dispatch nodes. When the average load is greater than 0.8, an instance is being added – when it is lower than 0.4 an instance is shut down. Spinning up a new instance typically takes around 3 min. In all deployments up to now, this infrastructure worked without any problems that would have affected the experience of the end user.

For the heat map sub-system used for receiving and aggregating the recorded data we relied on the CoenoSense system [19] developed at ETH Zurich's Wearable Computing Lab until mid 2013.

In June 2013 we began migrating apps to a custom made solution. Both systems have a sharded MongoDB[3] at their core which was designed for maximum performance and scalability from the beginning making it more suitable than alternatives such as CouchDB (which uses REST over HTTP compared to MongoDB's much quicker binary protocol between the DB server and the client application).

Compared to CoenoSense, the new solution relies on Lighttpd[4] as a web server which provides a higher performance. Another addition was the introduction of a queue layer. The data packets received by the web server are being passed on to a pool of input workers realized in the Python programming language. These workers write the data in a Beanstalkd[5] queue where it's being buffered. The data is then being processed by output workers which are

also written in Python. The output workers store the data in the sharded MongoDB. Benchmarks with 10,000 simulated concurrent users have shown that this solution for the backend reaches a request response time of around 0.04 s.

The application server in this sub-system hosting the web front end is explained in more detail in the following section.

5.3 Web front end

Both the administration and the heat map sub-systems offer an easy-to-use web interface to allow for a maximum compatibility with client devices. The interfaces have been tested with the latest builds of Firefox, Chrome and Safari. Therefore, they can be run on a multitude of device categories ranging from desktop PCs to tablets and even smartphones.

The administration front end is realized as a lightweight Rails application. More complex UI elements such as the map drawing tools and the calendar administration interfaces have been implemented using JavaScript and HTML 5 components. Each module's administration tool stores the user inputs into the SQL database. Simple modules trigger the creation of their corresponding JSON files after each change performed by the user. More complex modules (e.g. the App Builder or the Map Editor) have a "Publish" button that needs to be pressed by the user before the current content of the module is published as a JSON file.

The heat map web interface is realized using the Django framework which is written in Python and requires an additional SQL database (which has been omitted in Fig. 8 for reasons of clarity as it is unimportant to the heat map itself) to store its administrative data. The framework approach simplifies tasks such as user management, URL handling and session security greatly. Usually, heat map parameters such as initial zoom level etc. are being set automatically via a REST API accessed by the administration sub-system upon creating or editing a recording period. However, the heat map web interface allows for manual corrections of these values if the need should arise. As this part of the system is only accessed by administrators, it has been designed much more simplistic compared to the app administration system (see Fig. 9).

The heat map GUI itself requests the latest information about crowd conditions in regular intervals from the application server where it is stored in a MongoDB. When doing so, it sends a timestamp along with the request specifying which data it is interested in. The application server then fetches all the available location data from the Mongo DB cluster and returns a data structure that contains one location for each user that contributed data at that specific time.

The data is delivered by the application server to the GUI in a compact JSON format. Only when that data arrives at the GUI is it rendered. Offloading the heat map rendering task to the client (i.e. the GUI) saves considerable resources on the server side thereby making sure that the heat map can be accessed by many clients simultaneously. If the server was delivering rendered heat maps or tiles to the clients, it would need to be scaled much larger and thereby become much more expensive.

As mentioned earlier, the heat map view provides a time slider for inspecting past data. Technically, this has been implemented using an AJAX approach (a background call to the server) which allows for an almost instant displaying of the required data without a page refresh.

To provide a standard level of security against unauthorized access, the heat map is protected with a token only known to the administrative backend which seamlessly integrates the heat map view provided by the heat map sub-system's front end. However, it is worth noting that this view can also be accessed individually if the situation calls for it (e.g. if a police officer on the ground wants to access it on a tablet device). Therefore, the heat map

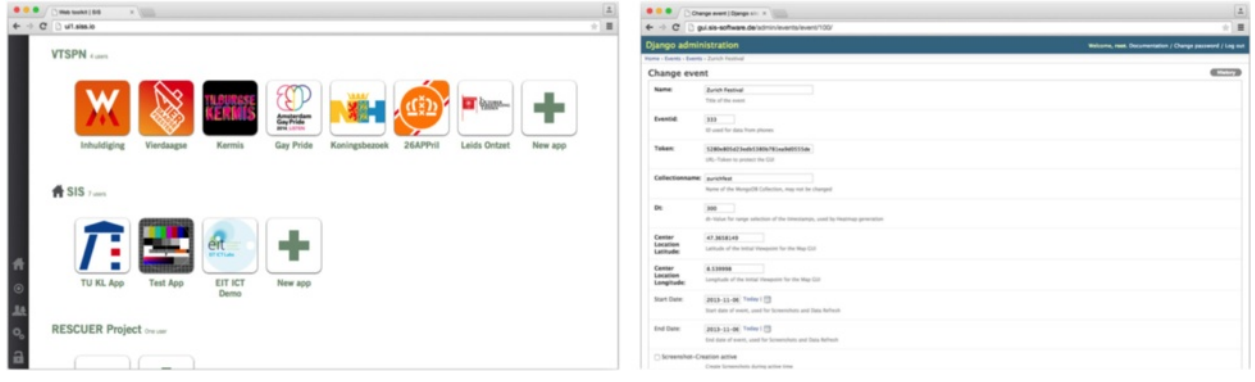

Fig. 9 Screenshots of the web front ends (*left*: app management, *right*: heat map configuration)

administration system can generate a URL which gives access to a webpage containing only the heat map without any other elements of the system.

6 Results and lessons learned

As mentioned above, our system has been deployed at a series of events throughout Europe and was used by well over 100,000 people. Since a more detailed description and evaluation of the crowd density estimation as it was performed in Chapter 3.1.1 goes beyond the scope of this system overview and experience paper – and has already been published in [13] – we want to focus on a broader description of the results of the deployments and report on the user feedback.

As Figs. 1 and 4 demonstrate, visualizing the crowd density in form of a heat map generated by crowd sourced real-time location data provides a good representation of the situation on the ground. Furthermore, the visualization gives a good insight into the temporal and spatial evolution of the crowd, thereby facilitating situational awareness, prediction and post event analysis.

One of the most important lessons learned during the numerous deployments was that it is absolutely vital to have a solid PR strategy for the distribution of the event apps. Chapter 3.1.1 clearly shows that the system performs well with only a few percent of the visitors sending data to our backend. Our system has shown to work reliably with a user penetration of around 1 or 2 % – a user penetration of 10 % would even allow for a very detailed analysis. However, depending on the size of the event, even a small percentage of app users can be a sizable absolute number. During our first deployments, visitors were only made aware of the app via social medial channels and the event website. The resulting number of downloads (see Table 3) based on this PR approach was not very satisfying. Over time we learned to better deal with this situation and during the deployment at the coronation of the Dutch King in 2013, roughly 10 % of the event visitors had downloaded the app (about 70,000). The PR campaign responsible for this success was employing numerous channels ranging from online articles, social media campaigns and print articles to actual TV coverage. As this might be more effort than some event organizers are willing to put into PR for an app, the Zurich festival proved that there are also simpler ways to achieve a solid number of downloads: on top of a social media campaign and a prominent download section on the event's website, the app was simply mentioned on all event posters and in event brochures. In summary, it can be said that the event apps should be made an integral part of the event planning process if the system is to deliver reliable results.

The relevance of the data recorded with our system is furthermore underlined by Fig. 10. It can be seen that the estimated crowd size correlates with real life events using the example of the 2013 Zurich festival. Furthermore, the figure demonstrates one of the system's weaknesses: network outages. The fireworks are always one of the highlights of the Zurich festival. During the fireworks on July 5, so many people gathered on the bridge facing the Lake Zurich, that parts of the cellphone network collapsed. This can be clearly seen in Fig. 10. Possible solutions for dealing with this problem are presented in the final section of this paper.

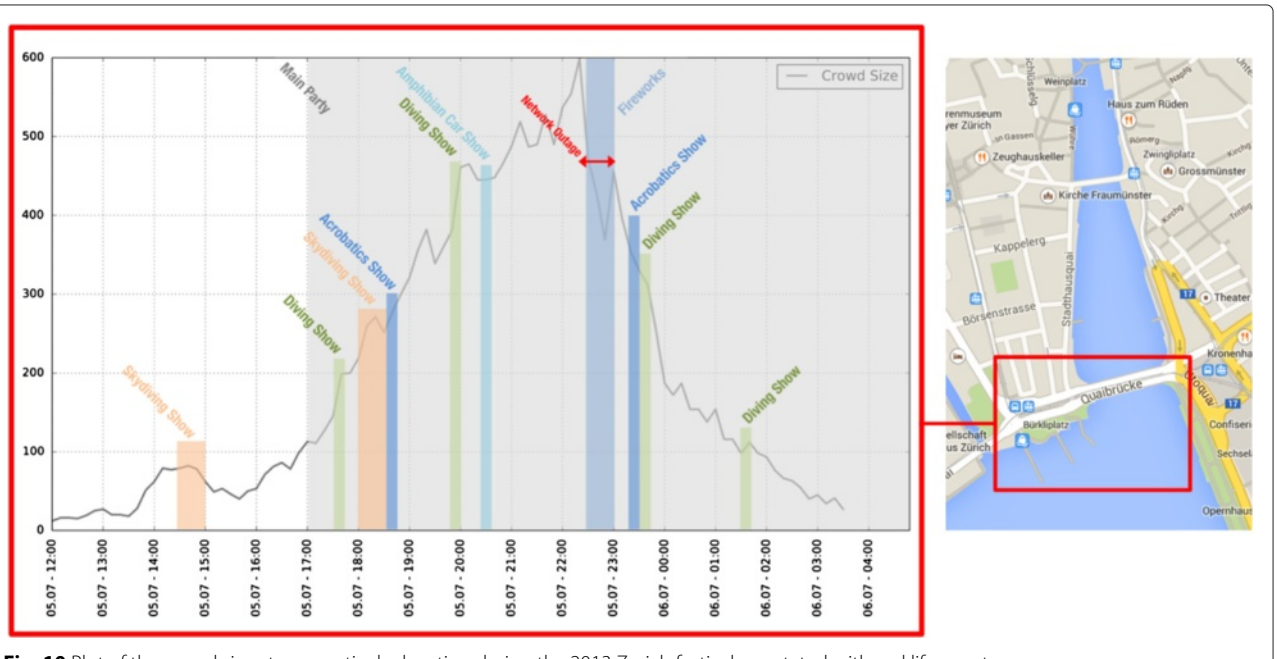

Fig. 10 Plot of the crowd size at one particular location during the 2013 Zurich festival annotated with real life events

How is the System perceived?

User Opinion

End Users

84 % Ranked the usefulness of the system as 'high' or 'very high'.

82 % Would actively consult the app in case of an emergency.

94 % Would follow the app's advice.

91 % Would only trust the app's advice if they knew the source of the information.

< 1 % Generally does not trust any advice given out on a phone.

83 % Initially downloaded the apps because of the available background information on an event and the map content.

79 % Told their friends about the app.

Source: Survey performed during some deployments of the system (622 participants).

App Usage

Feature Popularity

35 % Heat Map (in-app version)

16 % Map Content

15 % Twitter News

14 % Event Program

11 % Messages

Source: App usage logs sent by the apps during four deployments (410.141 total logs)

Operators

Peter Clarke (City of London Police):

Information on what happened after the event was particularly interesting.

The deployment of the system was the first time, the Police had knowledge about crowd movements during fireworks as CCTV doesn't work in the dark.

Interesting insights into how people moved around closed roads.

Brian Drayson (British Transport Police):

The system would be most welcome during an evacuation.

Some events like New Year's Eve are less organized. The system is especially useful in these instances as it's usually very hard to anticipate where people might move to.

Would like to identify day-to-day travel patterns and most common routes with the system.

Alison Ingleby (Greater London Authority):

Is most interested in the communication perspective and the ability to communicate during an evacuation.

Is very keen on exploring the possibility of a coordinated approach with multiple agencies feeding information into an app.

Inspector Peter Clarke from the City of London Police working with the system.

With respect to user acceptance of the system, we performed debriefing sessions with civil protection authorities and surveys with app users. While details about both can be found in the box "How is the System perceived?", it seems particularly worth mentioning that all participating authorities were keen on deploying the system at a larger scale and also wanted to explore further use cases for it. From an event management point of view, it is very encouraging to know that 82 % of all survey participants would actively consult such an app in case of an emergency, 94 % would follow the app's advice and only less than 1 % generally don't trust advice given out on a phone.

In the previous section we gave an overview of the system's architecture – in particular we described the generic app's main components. Amongst those components, the Statistics Manager was mentioned. This object's purpose goes a lot further than just sending information about how popular each feature of an app is. Instead, it collects information about how the app is being used and which requests have been made.

Specifically, it is being logged *what* feature is being used *where* and *when*. Also, for requests concerning points of interest (i.e. navigation requests or requests for more information about the item), it is being logged when and where the request was made, what was requested and where the requested item is located at. Evaluating this information on the system's backend gives an insight into what the crowd is currently interested in. This information in turn, can be used as an even earlier predictor for future crowd characteristics. For example, if a meaningful part of the crowd is interested in a certain concert at an event, the system could make an estimate about the growth of the crowd in front of the concert stage.

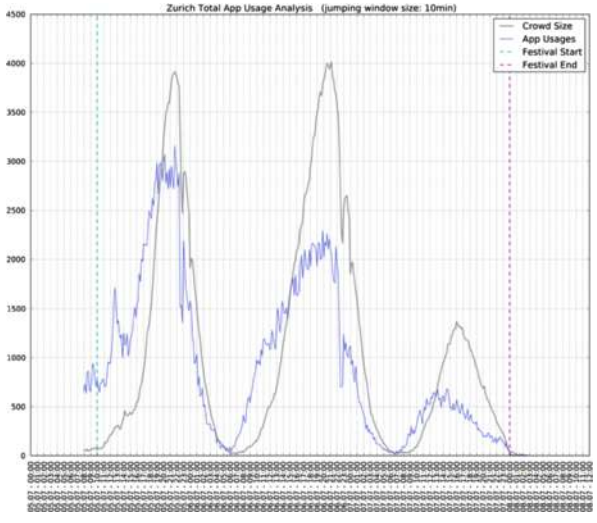

Fig. 11 Correlation between the size of the crowd and the app usage

Figure 11 visualizes this approach using again the example of the 2013 Zurich festival. The grey line represents the evolution of the size of the crowd as measured by our system – please note that for this particular event there was no ground truth available as we had no access to video footage and the event was also open to anybody, hence there was no information about the number of tickets available. The blue line represents the usage of the event app. It can be clearly seen that changes in the app's usage curve antedate corresponding changes in the curve of the crowd size.

We consider this representation of user interest an important factor for predicting crowd behavior. Future research will focus on further developing this approach.

7 Conclusion and future work

In this work we presented the evolution of a smartphone based crowd management system from a simple research prototype to a full blown event management solution which is currently being commercialized. The system was deployed at multiple large scale events throughout Europe and was used by several civil protection authorities – all of which highly valued the impact the system had on their work.

The app element of the system was downloaded by well over 100,000 people who collectively contributed over 100 million data points. The general public responded very well to the concept of event apps collecting anonymous data for the greater good.

We furthermore demonstrated the principal architecture of all system components, thereby presenting a system design for reconfigurable, scalable smartphone based crowd sourcing systems which could also be used for other purposes.

While our work's outcomes show that the system is generally up to its task, there are a number of issues which need to be addressed during future work:

One of the main problems we experienced is related to network connectivity. Complete network failures were never experienced during our deployments. However, during a New Year's Eve event, a 15 min blackout did occur in an isolated area. During those times, the crowd density heat map stayed empty in the affected area for obvious reasons. Also, this area couldn't be contacted with messages during the outage. In case of an emergency, such a network blackout could have drastic consequences.

Therefore, one focus of future work should be on the implementation of alternative means of communication in case of network blackouts. The most obvious solution seems to be the implementation of an AdHoc/Mesh-like opportunistic networking approach to bridge those areas without network connectivity.

Secondly, at one point our research work suffered from the interference of a telecommunication provider who claimed that the network wouldn't support the traffic caused by our app. While those arguments could be countered eventually, it still proved the point that we failed to include telecommunication providers as an important stakeholder in our concept. We're therefore currently implementing means for providers to define critical thresholds for their infrastructure which will not be exceeded by our system.

Thirdly, future versions of our system will take feedback we received from civil protection authorities into account. For example, GLA (Greater London Authority) expressed the wish to have multiple agencies feed contents into the same system so that it would allow for an integrated workflow.

Finally, the system will be used to establish a living lab at the Technical University of Kaiserslautern. Therefore, a university app is currently being created which is based on our framework. The goal is to establish a platform that will enable research groups to run large scale experiments using crowd sourced data from participating students.

Endnotes

[1]http://www.nottebiancamalta.com
[2]http://www.lordmayorsshow.org
[3]http://www.mongodb.org
[4]http://www.lighttpd.net
[5]http://kr.github.io/beanstalkd/

Competing interests
The authors confirm that they have read SpringerOpen's guidance on competing interests and state that none of them have any competing interests in the manuscript.

Authors' contributions
TF designed the system under the supervision of PL and carried out most of the deployments. UB organized and carried out the system's deployment at the 2013 Zurich festival and contributed to data analysis. TF and PL wrote the manuscript. All authors read and approved the final manuscript.

Acknowledgements
The authors would like to thank all student helpers – especially Torben Schnuchel – who contributed to the creation of the system with countless hours of programming work. The initial steps of this work were supported by the Socionical project, funded under the European Commission's FP7 program (grant: 231288). Further funding was received from the European Commission's FP7 program under grant agreement #600854 "Smart Society - hybrid and diversity-aware collective adaptive systems: where people meet machines to build smarter societies" and by the CoCoRec (Collaborative Context Recognition in Dynamic, Multimodal Smart Environments) project supported by the German Federal Ministry of Education and Research.

Author details
[1]German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany. [2]ETH Zurich, Zurich, Switzerland.

Received: 26 May 2015 Accepted: 5 November 2015
Published online: 22 December 2015

References

1. Rahmalan H, Nixon MS, Carter JN. On crowd density estimation for surveillance: IET; 2006. http://digitallibrary.theiet.org/content/conferences/10.1049/ic_20060360.
2. Davies AC, Yin JH, Velastin SA. Crowd monitoring using image processing. Electron Commun Eng J. 1995;7(1):37–47.
3. Lo B, Velastin S. Automatic congestion detection system for underground platforms. In: Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on. IEEE; 2001. p. 158–61.
4. Chan AB, Liang Z-S, Vasconcelos N. Privacy preserving crowd monitoring: Counting people without people models or tracking. In: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on. IEEE; 2008. p. 1–7.
5. Murakami Y, Minami K, Kawasoe T, Ishida T. Multi-agent simulation for crisis management. In: Knowledge Media Networking, 2002. Proceedings. IEEE Workshop on. IEEE; 2002. p. 135–9.
6. Shendarkar A, Vasudevan K, Lee S, Son Y-J. Crowd simulation for emergency response using bdi agent based on virtual reality. In: Proceedings of the 38th conference on Winter simulation, Winter Simulation Conference. IEEE; 2006. p. 545–53.
7. Kluepfel HL. A cellular automaton model for crowd movement and egress simulation. Fakultät für Physik: PhD thesis, Universität Duisburg-Essen; 2003.
8. Asimakopoulou E, Bessis N. Buildings and crowds: Forming smart cities for more effective disaster management. In: Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011, Fifth International Conference on. IEEE; 2011. p. 229–34.
9. Roitman H, Mamou J, Mehta S, Satt A, Subramaniam L. Harnessing the crowds for smart city sensing. In: Proceedings of the 1st international workshop on Multimodal crowd sensing. ACM; 2012. p. 17–8.
10. Szabo R, Farkas K, Ispany M, Benczúr A, Batfai N, Jeszenszky P, et al. Framework for smart city applications based on participatory sensing. In: Cognitive Infocommunications (CogInfoCom), 2013,IEEE 4th International Conference on. IEEE; 2013. p. 295–300.
11. Ghose A, Biswas P, Bhaumik C, Sharma M, Pal A, Jha A. Road condition monitoring and alert application: Using in-vehicle smartphone as internet-connected sensor. In: ervasive Computing and Communications Workshops (PERCOM Workshops), 2012, IEEE International Conference on. IEEE; 2012. p. 489–91.
12. Pan B, Zheng Y, Wilkie D, Shahabi C. Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM; 2013. p. 344–53.
13. Wirz M, Franke T, Roggen D, Mitleton-Kelly E, Lukowicz P, Troster G. Inferring crowd conditions from pedestrians' location traces for real-time crowd monitoring during city-scale mass gatherings. In: Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2012, IEEE 21st International Workshop on. IEEE; 2012. p. 367–72.
14. Wirz M, Franke T, Roggen D, MitletonKelly E, Lukowicz P, Tröster G. Probing crowd density through smartphones in city-scale mass gatherings. EPJ Data Sci. 2013;2(1):1–24.
15. Helbing D, Buzna L, Johansson A, Werner T. Self-organized pedestrian crowd dynamics: Experiments, simulations, and design solutions. Transport. Sci. 2005;39(1):1–24.
16. Helbing D, Farkas IJ, Molnar P, Vicsek T. Simulation of pedestrian crowds in normal and evacuation situations. Pedestrian Evacuation Dyn. 2002;21: 21–58.
17. Ngai KM, Burkle FM, Hsu A, Hsu EB. Human stampedes: a systematic review of historical and peer-reviewed sources. Disaster Med Publ Health Preparedness. 2009;3(04):191–5.
18. Helbing D, Mukerji P. Crowd disasters as systemic failures: analysis of the love parade disaster. EPJ Data Sci. 2012;1(1):1–40.
19. Wirz M, Franke T, Mitleton-Kelly E, Roggen D, Lukowicz P, Tröster G. Coenosense: A framework for real-time detection and visualization of collective behaviors in human crowds by tracking mobile devices. In: Proceedings of the European Conference on Complex Systems 2012. Springer; 2013. p. 353–61.
20. Franke T, Lukowicz P, Wirz M, Mitleton-Kelly E. Participatory sensing and crowd management in public spaces. In: Proceeding of the 11th annual international conference on Mobile systems, applications, and services. ACM; 2013. p. 485–6.

21. Blanke U, Troster G, Franke T, Lukowicz P. Capturing crowd dynamics at large scale events using participatory gps-localization. In: Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014, IEEE Ninth International Conference on. IEEE; 2014. p. 1–7. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6827652&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel7%2F6820824%2F6827478%2F06827652.pdf%3Farnumber%3D6827652.
22. Oberhagemann D. Statische und dynamische Personendichten bei Grossveranstaltungen. Technical Report of the Association for the Improvement of German Fire Protection TB 13-01. 2012. http://www.vfdb.de/download/TB_13_01_Grossveranstaltungen.pdf.

Submit your manuscript to a SpringerOpen journal and benefit from:

► Convenient online submission
► Rigorous peer review
► Immediate publication on acceptance
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

Submit your next manuscript at ► springeropen.com