CrumblR: Enabling Proxemic Services through Opportunistic Location Sharing

Geert Vanderhulst, Marzieh Dashti*, Afra Mashhadi*, and Fahim Kawsar
Bell Laboratories
Alcatel-Lucent
Antwerp, Belgium
* Dublin, Ireland
{firstname.surname}@alcatel-lucent.com

***Abstract*—The need for a device to determine its own location and share it with individual Location-Based Services (LBSs) prevents large-scale adoption of *indoor* services, specially those belonging to small businesses or those intrinsic to short-lived events. We present CrumblR – a proxemic service broker platform that advocates anonymised location sharing with place owners instead of LBS providers to receive value-added contextual services. By decoupling location detection from LBSs, CrumblR liberates users from disclosing their locations to individual service providers. Our platform relies on opportunistic sharing of wireless signal fingerprints with nearby places to enable discovery of *proxemic* services associated with the place. In this WiP paper, we present the architectural design and the simple place proximity detection algorithms used in CrumblR.**

I. Introduction

Today, the location of mobile devices is massively shared and exchanged to facilitate a variety of Location-Based Services (LBSs). Ranging from public social LBSs such as Foursquare[1] to private enterprise ones such as Loc-aid[2], LBSs have truly taken advantage of the proliferation of GPS-enabled devices. The prevalent LBSs are based on a fundamental characteristic that couples Location and Services together; that is each LBS builds its own database of locations alongside other features (such as ratings, reviews, etc). This coupling enforces the end-user to subscribe to each LBS individually by downloading smartphone applications, in order to take advantage of services that might be available to her at a specific location. Although such model may work for popular applications providing substantial LBSs (e.g., Foursquare), it is less ideal for services belonging to small businesses or those intrinsic to short-lived ad-hoc events. Furthermore, a subscriber requires to determine its own location and share it with each LBS separately. However, due to poor GPS signal reception, a device often fails to reliably locate itself in most public and private places such as shopping malls and enterprise buildings. To cater for this scarcity, LBSs often have to further rely on proprietorial expensive sources of data such as those collected by Google, Apple, Skyhook Wireless, which still have limited indoor availability and accuracy.

Recently many research studies [1], [8], [9] and services such as Apple Indoor Positioning have leveraged the WiFi modules incorporated in smartphones to determine a user's indoor location. This method, based on WiFi fingerprinting, relies on collecting and matching the fingerprints of location-dependent characteristics, such as the received signal strength from nearby WiFi Access Points (APs). The fingerprints obtained at different locations are then semantically labelled and stored in a database associated with their coordinates. This allows for any newly collected fingerprints to be compared with the database and thus infer the indoor location of the users. However, one bottleneck of such approach involves the collection of these rich semantically labelled fingerprint databases which dynamically change over time.

In this work, we argue that by decoupling the location detection from the LBSs, we are able to provide both the end-users and service providers with a more dynamic base for *Proxemic Services* in indoor spaces. We define a Proxemic Service as a "temporal service that automatically provides the user with value at a specific place". Unlike a traditional LBS, devices do not share their location with individual proxemic services but rather with the place or spatial entity that the service is associated with. Borrowing our terminology from the story of Hansel and Gretel, we define a *crumb* as an anonymous wireless signal fingerprint that captures both cellular and WiFi signal sources. When residing at a trusted place, a device starts sharing crumbs anonymously with the place. Similar to Hansel's bread crumbs, signal crumbs do not give away the identity of a user, nor can they be spotted outside the place (i.e. by a service that does not maintain a relationship with the place). For instance, by "checking in" to an airport, crumbs are shared with the airport such that the user gains access to all proxemic services associated with the airport – e.g., location-based notification services such as "The Starbucks on your left offers 15% discount on all pastries", "Gate change: the flight from gate B2 will depart from gate B10", "Airport security warns for long queues near the A gates.". Whilst the stakeholders of these services clearly differ, they leverage the same crumbs and provide the user with a unified experience at a given place.

The key benefits of our platform, called CrumblR, are twofold: (i) a per-place subscription model rather than a per-service subscription model allows new services associated with a place to opportunistically reach the end-users and (ii)

[1]https://foursquare.com/
[2]http://www.loc-aid.com/

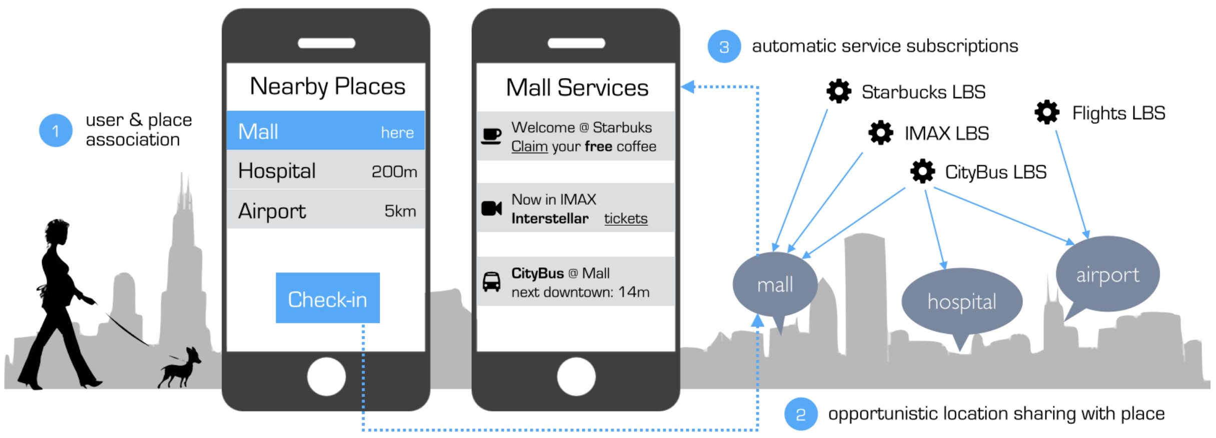

Fig. 1. CrumblR lets users check-in to a place (1), share their location context with multiple proxemic services at once (2) and delivers a unified experience to the end-user by only presenting those services relevant at a place (3).

improved efficiency in terms of energy consumption on the device, as the user only needs to share her location once to take advantage of many services available to her.

II. Design and Architecture

CrumblR associates *places* with *services*. A place in CrumblR is similar to a place in Foursquare: it can be a corporate building, a public facility, a small business, etc. Since we perceive that many LBSs only make sense at particular locations, we focus on exactly those proxemic services which offer a value to the user whilst residing at a place, e.g., critical notifications, loyalty discounts, personalised navigation, etc. Figure 1 depicts the CrumblR prototype application. First, the user is presented with an overview of places known to CrumblR near her current location. By checking in to a place, her device starts dropping crumbs – wireless signal fingerprints – at the place.[3] In return for opportunistically sharing location hints with a place, proxemic services associated with the place push content to a user's device (e.g., coupons, alerts, interactive controls, etc). When leaving the place, the user is automatically unsubscribed from these services which can no longer track her.

A place is registered on our platform with a geographic coordinate and a list of "fingerprinted points". This list consists of wireless signal measurements near a Point-in-Place (PiP) that is of particular interest for that place, e.g., a gate, security counter, … at an airport. The geographic coordinate allows CrumblR to list nearby places on the user's device via GPS, whereas the fingerprinted points allow CrumblR to identify whether a user is co-located with a PiP (section III-B). To integrate with CrumblR, a service should implement an input channel (i.e., a REST interface) to receive locations from places, and an output channel to push content (i.e., active HTML snippets) to a user's device. The owner of a registered place on CrumblR (who does not necessarily need to own the infrastructure of the place), can then associate registered services with her venue. CrumblR acts as a broker platform associating users and services seamlessly and anonymously across trusted places. Figure 2 depicts the situation where

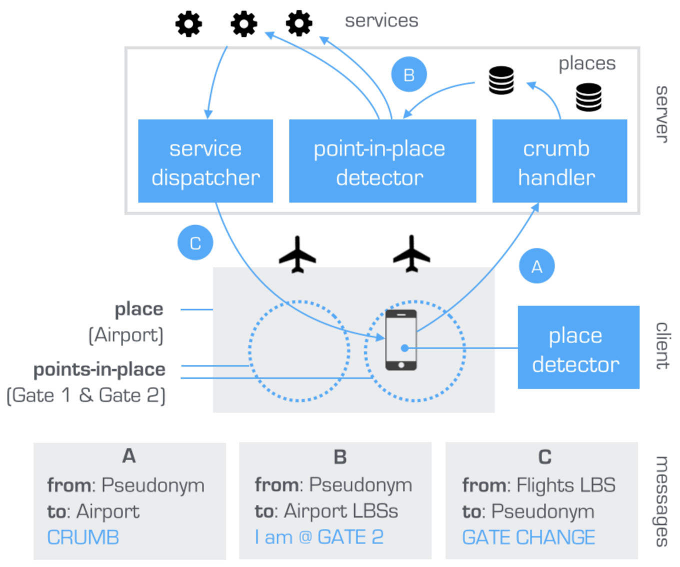

Fig. 2. Interaction between different components in CrumblR.

a user has checked in to a place (airport), shares a crumb with the place and gets associated with a service based on her presence at a PiP within the place. To protect the user's privacy, we represent users by anonymous pseudonyms in their communication with CrumblR such that a user's identity is never revealed to either CrumblR or individual services. Via the crumb handler, a crumb arrives at the place's instance on CrumblR from where it is forwarded to the PiP detector. Given the airport's list of fingerprinted points, the PiP detector determines groups of users residing near known PiPs. For instance, if one or more users are detected to be co-located with a PiP labeled "Gate 2", the pseudonyms of these users and PiP label are propagated to the airport's associated services.

[3]By default, these crumbs are generated every minute to conserve energy. However, we envision an adaptive sampling rate that scales up and down based on a place's requirements.

Via the service dispatcher, a proxemic service reaches out to all users located near Gate 2 to notify them of a gate change.

III. Algorithms and Preliminary Results

CrumblR uses two algorithms to determine the user's location: (i) a *place detection* algorithm which is used for automatically checking in to previously trusted places by detecting a device's coarse location (e.g., airport) and (ii) a *point-in-place detection* algorithm to detect a device's location with finer granularity (e.g., specific airport gates). The former runs as an offline algorithm on the end-users' devices without the need for a network connection, while the latter is an online algorithm which is executed once the device starts sending crumbs to a recognised place.

A. Place Detection via Minimal Radio Maps

Users' preferences and trust in a place has shown to vary over time and change dynamically depending on their social context. Various works have previously examined ways for addressing this dynamic change [7], [5], [2]. In this work our focus is on scenarios where the users' trust in a place is persistent, however we believe similar techniques could be incorporated into CrumblR. To this end, we designed a mechanism to support automatic check-ins in trusted places, similar to a phone automatically connecting to a trusted WiFi AP. This is achieved by relying on two fundamental operations of mobile phones: cellular and WiFi probing, enabling a phone to learn about the identities of cell towers and WiFi APs within radio range.

First, we maintain a list of places to which the device has previously sent crumbs, i.e., places the user has checked-in previously. Each place stores a Minimal Radio Map (MRM) that consists of all perceived cell IDs and WiFi (B)SSIDs at that place. This MRM is extracted from the list of fingerprinted POIs sent to CrumblR when registering a place and it is stored on a user's device after checking in to the place. Our place detection algorithm then matches cell nodes and WiFi APs currently in sight by a device with those stored in MRMs. After a cell look-up, cell IDs are compared with those perceived at each place, and a place is flagged as potentially "nearby" in case of a match. This operation carries low overhead in terms of energy consumption as cell look-ups are performed continuously to maintain a connection with a mobile operator.

Next, if one or more places have been flagged but no WiFi connection is established, we rely on WiFi probes to discover APs in the vicinity. Each newly discovered AP in a probe is then compared with the flagged places' MRM, and in case of a match, the device automatically checks in at the place and starts sending crumbs. It should be noted that the accuracy of our place detection is limited by the range of an AP. For instance, if two neighbouring shops are both known as trusted places, they might both be detected and crumbs will be sent to each shop. To identify the more precise location of the user, we combine the place detection algorithm with the point-in-place detection algorithm which we describe next.

Finally, to further reduce the power usage of our place detection algorithm, we limit the number of WiFi probes and use the probing mechanism (which does not require a device to connect with an AP) only when the user has no active connection with a known WiFi AP. This approach could be extended by incorporating other sources of information to infer users' displacement such as monitoring inertial sensors [8].

B. Point-in-Place Detection based on Co-Location

We address location detection within a place as a co-location problem, i.e., identifying which users are co-located with known PiPs. We first identify PiPs at a place (e.g., shops in a mall, gates at an airport, etc) and collect an RF fingerprint at (the center of) each PiP. Such RF fingerprint is a vector of available AP IDs and their associated RF measurements, i.e., received signal strength (RSS) measurements in our case. The idea behind our co-location technique is that the multipath structure of a radio channel is unique to every location and can be considered as a signature of the location. Co-located devices experience a similar multipath environment (e.g., reflectors and objects in the environment) and therefore exhibit similar multipath profiles. By combining live fingerprints (from user devices) and a small set of pre-recorded fingerprints (from PiPs), we link groups of users and PiPs with minimal fingerprinting effort. In our approach, it is sufficient to collect fingerprints only at PiPs – site-surveying an entire floorplan or training a localisation algorithm is not required.

Our co-location algorithm calculates the signal distance between every two nodes (user device nodes and PiP nodes). Different distance metrics can be applied to measure the distance between two fingerprint vectors. The simplest metric is the Manhattan distance in which the sum of the absolute differences of signal strengths is computed. Assume that r_i^A and r_i^B denote RSS values from the i-th AP observed by user A and B respectively. Also assuming that N APs are seen by two nodes A and B, their Manhathan distance [3] in the signal space can be calculated as

$$d_{A,B} = \sum_{i=1}^{N} |r_i^A - r_i^B| \tag{1}$$

The nodes whose fingerprints differ less than a dissimilarity threshold δ are considered potentially co-located. In order to evaluate the accuracy of our approach, we construct a connectivity graph of nodes based on fingerprint similarities. If the distance $d_{j,k}$ between a j-th and a k-th node is less than δ, the two nodes are connected by an edge in the graph, i.e., $C_{j,k} = 1$. Otherwise, two nodes are disconnected, i.e., $C_{j,k} = 0$. Out of this graph we can easily extract groups of co-located nodes. If such group contains a PiP, we know that the users' devices in that group reside near the PiP. This information is then communicated to the services associated with a place.

We tested our approach inside an enterprise building at a test area spanning 2500 m^2 and consisting of 27 office cubicles, 16 meeting rooms and corridors. First, we collected PiP

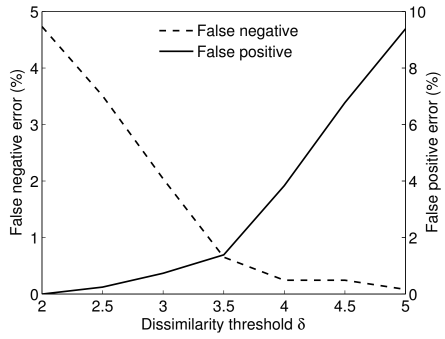

Fig. 3. Analysis of dissimilarity treshold for a 50 node connectivity graph.

fingerprints near the center of each cubicle and meeting room and registered the place on CrumblR with this information. Next, we provided 5 participants with smartphones running a prototype of CrumblR. This application continuously sent crumbs (RF fingerprints) to the place's instance on CrumblR from where they are forwarded to the PiP detector (Figure 2). We then dispersed our participants over the test area. As our number of participants is limited, we repeated this process 10 times. We then extracted the time dimension and aggregated the data to resemble a 50 node connectivity graph. Since the purpose of this preliminary evaluation is to analyse the accuracy of our clustering approach, we choose not to distinguish between user nodes and PiP nodes. In this case study, we considered two nodes to be co-located when there is less than 2 meters distance between them. To find the optimal dissimilarity treshold, we evaluated the algorithm's performance for different values of δ using a connectivity error metric. If two nodes j and k were in reality co-located, we define $C'_{j,k} = 1$ indicating a true link between these two nodes, otherwise $C'_{j,k} = 0$. We calculate $e_{j,k} = C_{j,k} - C'_{j,k}$ for every two pairs of nodes. The connectivity error $e_{j,k} = -1$ indicates a false negative error and $e_{j,k} = 1$ indicates a false positive error. Figure 3 shows the impact of δ on connectivity errors for a normalised Manhattan distance metric. Our early results indicate the δ of our test area to be 3.5. This results in less than 1 % false negative and less than 2 % false positive connectivity errors for a connectivity graph of 50 nodes.

IV. Discussion

In CrumblR, users voluntarily and anonymously share their locations with a place to obtain associated proxemic services. However, sometimes these services might overwelm a user with information she is not interested in. We believe this problem can be addressed via folksonomy-based profiling such as those proposed in the publish/subscribe paradigm [6]. Besides, services lack personalisation options since they are not aware of the user's identiy. We envision proxemic services to request permissions, similar to mobile applications, to access certain types of information like the user's identity and hence unlock personalisation features when approved by the user.

To detect groups of users near PiPs, we rely on basic WiFi fingerprinting techniques. However, another solution would be to leverage emerging BLE technologies such as Estimote beacons[4] where each PiP can be represented by a beacon. Still, the issue with such beacons is their associated deployment and management cost. Place owners would need to invest to make their venue compatible with CrumblR which is not the case with our present approach where existing infrastructure is used. However, once BLE-based beacons become widely adopted, our platform can be tuned to take advantage of them when available.

As part of our future work we plan to evaluate the performance of CrumblR more broadly, namely in scenarios such as shopping malls where the number and distribution of APs can vary greatly. Such deployments would allow us to evaluate the complexity of CrumblR and its algorithms as the scale increases. Furthermore, we plan to investigate how we can verify crumbs, catering for scenarios where a user has an incentive to lie about her whereabouts. For example, a user might replicate a previously recorded crumb to claim she resides at a given place while she is not. This problem of proving one's location has been studied before [4]. Due to the nature of our approach – opportunistically sharing crumbs – we see an opportunity to seamlessly acknowledge each other's crumbs via crowd-sourcing.

Finally, we envision CrumblR to act as a *collective analytics platform* where statistical knowledge about spatial places is provided to interested parties (e.g., place owners). For instance, the crumbs provided by users at a mall can provide insights about the number of visitors at the mall at different times, the most popular stores, etc.

References

[1] Jacob T Biehl, Matthew Cooper, Gerry Filby, and Sven Kratz. LoCo: a Ready-to-Deploy Framework for Efficient Room Localization using Wi-Fi. In *Proc. of UbiComp*, pages 183–187, 2014.

[2] Greg Bigwood, Fehmi Ben Abdesslem, and Tristan Henderson. Predicting Location-sharing Privacy Preferences in Social Network Applications. *Proc. of AwareCast*, 2012.

[3] Sung-Hyuk Cha. Comprehensive Survey on Distance/similarity Measures between Probability Density Functions. *Int. Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307, 2007.

[4] Wanying Luo and Urs Hengartner. VeriPlace: a Privacy-Aware Location Proof Architecture. In *Proc. of GIS*, pages 23–32, 2010.

[5] Norman Sadeh, Jason Hong, Lorrie Cranor, Ian Fette, Patrick Kelley, Madhu Prabaker, and Jinghai Rao. Understanding and Capturing People's Privacy Policies in a Mobile Social Networking Application. *Personal and Ubiquitous Computing*, 13(6):401–412, 2009.

[6] Mohammad Sadoghi and H-A Jacobsen. Relevance matters: Capitalizing on less (top-k matching in publish/subscribe). In *Proc. of ICDE*, pages 786–797. IEEE, 2012.

[7] Yasushi Sakura. How well can a user's location privacy preferences be determined without using gps location data. *IEEE Transactions on Emerging Topics in Computing*, 10(1):1, 2014.

[8] He Wang, Souvik Sen, Ahmed Elgohary, Moustafa Farid, Moustafa Youssef, and Romit Roy Choudhury. No Need to War-drive: Unsupervised Indoor Localization. In *Proc. of MobiSys*, pages 197–210, 2012.

[9] Zheng Yang, Chenshu Wu, and Yunhao Liu. Locating in fingerprint space: Wireless indoor localization with little human intervention. In *Proc. of MobiCom*, pages 269–280, 2012.

[4]http://estimote.com/