Björn Richerzhagen, Nils Richerzhagen, Julian Zobel, Sophie Schönherr, Boris Koldehofe and Ralf Steinmetz.
Seamless Transitions Between Filter schemes for Location-based Mobile Applications.
In Proc. 41st IEEE Conference on Local Computer Networks (LCN). ISBN: 978-1-5090-2054-6

Seamless Transitions Between Filter Schemes for Location-based Mobile Applications

Björn Richerzhagen, Nils Richerzhagen, Julian Zobel, Sophie Schönherr, Boris Koldehofe and Ralf Steinmetz
Multimedia Communications Lab (KOM), Technische Universität Darmstadt, Germany
E-Mail: {firstname.lastname}@kom.tu-darmstadt.de

***Abstract*—With a plethora of sensors and ubiquitous access to the Internet, modern smartphones have enabled a broad range of context-based applications. Most applications make use of the user's physical location to filter relevant content. However, filtering based on dynamic contextual information results in high complexity of the filtering process. This limits the applicability of existing publish/subscribe systems, as they rely on aggregation of filters and fast decentralized matching and forwarding. In this work, we propose a mechanism for transitions between different filter schemes for location-based services. Our mechanism adapts the filtering process to the dynamics in user behavior and resulting load by trading computational complexity at the broker against communication overhead and computational complexity at the mobile client. We integrate our mechanism into an existing publish/subscribe system and evaluate transitions between a context-based filter scheme and two channel-based filter schemes, showing the applicability of our approach.**

***Index Terms*—Location-based applications, mobile applications, publish/subscribe**

I. Introduction

Ubiquitous access to the Internet enables a broad range of cloud-backed applications on mobile devices. The vast majority of these applications utilize contextual information, most prominently the physical location of a client, to determine and filter relevant content. Examples range from mobile social networking applications and messengers to fully-fledged augmented reality games, focusing on the interaction between clients rather than on simple information retrieval. For such dynamic applications, matching user-generated content to interested consumers based on their location is usually done by brokers in the cloud, relying on the publish/subscribe communication paradigm. Location-based mobile applications require brokers to maintain and utilize context information provided by the clients during the filtering process. Although many systems for location-based publish/subscribe have been proposed by the research community [5], [9], their complexity when dealing with dynamic context information and the resulting limitations when considering distributed filtering often hamper their practical use. Furthermore, the efficiency of the proposed approaches is limited by the frequency of context updates, as efficient filter aggregation becomes increasingly complex in the face of dynamic subscriptions [9].

The performance of the proposed filter schemes strongly depend on the characteristics of the scenario, with some filter schemes being specifically designed for, e.g., vehicular movement on a road [12], [23] or pedestrian movement in a crowd [2]. These characteristics are highly dynamic for the case of location-based mobile applications, as they directly depend on the movement and interaction patterns of humans. In addition, the load on the system can vary significantly during rush hours or on specific events [21].

In this work, we propose a method for executing transitions between different filter schemes during runtime, rather than deploying a single, inflexible filter scheme. We focus on the protocol between a mobile client and its associated publish/subscribe broker rather than on the protocol used within the (potential) network of brokers. Most notably, executing transitions between filter schemes assists in trading complexity at the broker against resource utilization of the mobile clients depending on the current conditions and the desired trade-off between performance and cost. Based on this method, we propose a system design that enables the integration of transition-enabled filter schemes into a location-based publish/subscribe middleware. Our design includes generic means for state transfer in-between different schemes to enable seamless transitions and for reducing the coordination overhead during a transition. We implement a prototype of the proposed design featuring a number of filter schemes and evaluate the impact of transitions on the achievable performance under dynamic load conditions. Through an extensive simulation study, we show that our design achieves seamless switches between a context-based filter scheme derived from [9] and two channel-based filter schemes inspired by a grid-based geocast protocol [10]. Our state transfer mechanism ensures flawless delivery of events to interested clients during the execution of transitions.

The following contributions are presented in this work: (i) a method and system design for seamless transitions between filter schemes, with a focus on state transfer in-between different schemes, (ii) a prototype of the proposed design as an extension to an existing publish/subscribe middleware, and (iii) a simulative evaluation of the impact of filter scheme transitions on the performance and cost tradeoffs of the resulting system for the scenario of a mobile augmented reality game.

The remainder of this paper is structured as follows. We discuss the scenario of location-based mobile applications in Section II. In Section III, we present the method and design to support transitions between filter schemes and discuss our prototypical implementation. Section IV contains an in-depth simulative evaluation of our prototype, highlighting the impact of transitions on the overall system performance. After a discussion of relevant related work in Section V, Section VI concludes our paper.

II. Scenario

We focus on the scenario of location-based mobile applications that rely on a cloud-based infrastructure, as illustrated in Figure 1. This is the most common deployment model for mobile applications [11], [27]. Mobile clients access an application provider's service through a dedicated application on their mobile device. There exists a plethora of location-based applications, ranging from simple messaging applications to mobile social networks and fully-fledged augmented reality games such as Google's Ingress. A client can interact with the respective application in two different ways: either explicitly by using the application's controls or implicitly by changing relevant context, e.g. in consequence of movement.

These interactions trigger events that are then sent via a publish/subscribe middleware to the backend service of the application provider, which runs on a cloud infrastructure. There, events are filtered and forwarded to interested subscribers by a single broker, or a network of brokers. Clients are mostly interested in events that are associated to their close proximity (e.g., other player's actions in the case of a mobile augmented reality game), leading to a high locality of interest [25], [26]. Therefore, filtering is based on a mixture of static attributes – e.g., a topic or a set of categories – and contextual information such as the client's current location, depending on the utilized filter scheme. To support mobile clients and location-based subscriptions, different filter schemes have been proposed in the literature (c.f. Section V). They have in common that mobile clients need to update their subscriptions – or the associated contextual information – at the broker, depending on the required granularity of the respective filter scheme.

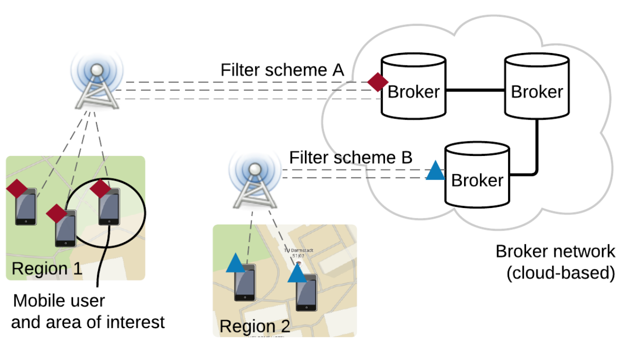


Fig. 1. The location-based application runs on mobile devices that are associated to a broker in the cloud-based backend of the server. Users are interested in events within a given area of interest around their current location.

Individual filter schemes exhibit different performance and cost tradeoffs with respect to their accuracy, their computational complexity, and their network utilization. In general, computational complexity at the broker (i.e., for the application provider), which is required for accurate filtering, can be traded against higher network utilization and increased complexity for mobile clients. The characteristics of individual filter schemes further depend on the load on the publish/subscribe middleware, and thereby on the behavior of the mobile clients. As a consequence, load can vary significantly throughout the day and for different locations [1]. To allow the middleware to adapt to such dynamic conditions, we propose to execute transitions between different filter schemes depending on the current load conditions and the desired performance and cost trade-off of the individual stakeholders. Transitions affect the filter scheme that is utilized between clients and their associated broker, as illustrated in Figure 1. We do not alter the filter scheme and associated routing protocol utilized within the broker network, as we assume that clients are already assigned to a broker based on their approximate location [22], [24]. Additionally, for scalability reasons, the filter scheme used within the broker network should not depend on an individual client's context, but operate on rather static attributes instead (e.g., a *region*, c.f. Figure 1).

Our proposed method for transitions between filter schemes and the resulting system design and prototype is described in the following section.

III. Design

To enable transitions between different filter schemes for location-based publish/subscribe, we need to encapsulate their functionality behind a common programming interface (c.f. Section III-A). Transitions are realized as local handovers between individual filter schemes by defining and controlling a common lifecycle for all filter schemes. However, as filter schemes rely on state information, it is vital to ensure correct conversion and transfer of state from one filter scheme to another (c.f. Section III-B). We integrate the proposed design into an existing publish/subscribe system for location-based services [18] and include a representative set of filter schemes from the literature (c.f. Section III-C).

A. Encapsulating Location-based Filter schemes

To enable transparent transitions between filter schemes, all associated functionality needs to be encapsulated behind a common programming interface. An overview of the resulting architecture is shown in Figure 2. Filter schemes consist of one component running on the broker that is responsible for maintaining and matching client subscriptions, and another component that provides a programming interface to the application running on the mobile clients.

On the client side, we extend the core API for publish/subscribe systems derived by Pietzuch et al. [16] to enable applications to issue location-based subscriptions. We utilize the concept of location requests, as defined in the Android-API. A location request is created by the application to define the frequency and desired accuracy of location updates. To create a subscription for content that is relevant in close proximity to the client's current position, the application passes the corresponding location request and a maximum distance to the *subscribe* method.

sub_handle **subscribe**(filter, callback, location_request, distance)

The location request is used by the filter scheme to register a listener for position changes using the mobile operating system's API. The respective filter scheme requires logic

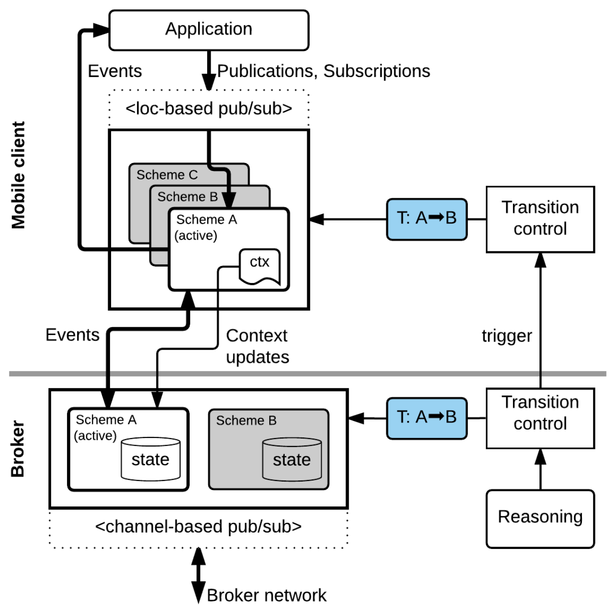


Fig. 2. Overview of components on mobile clients and the cloud-based broker. Publications and subscriptions are issued using the currently active filter scheme at the client through a unified API for location-based publish/subscribe. Besides forwarding and filtering events, the respective filter schemes define custom protocols for context information updates. Transitions interact with the lifecycle of filter schemes and the state information at the broker.

to update subscriptions at the broker – a simple approach would be to periodically send location update messages. These updates can either be sent as separate messages or they can be piggybacked to outgoing events to reduce communication overhead. Depending on the filter scheme, events published by clients are automatically tagged with the client's position upon invocation of the *publish* method. If an application explicitly wants to publish content to another geographic region, an extended *publish* method provides the ability to specify a target location.

```
void publish(event, target_location)
```

On the broker side, filter schemes need to provide a method to determine the set of clients that are to be notified of an incoming event. Each filter scheme implements a *get_subscribers* method that operates on the incoming message which carries the publication.

```
contacts[] get_subscribers(publication_msg)
```

The *publication_msg* carries the *event* published by the client, including the *target_location*, if provided by the client. The client's filter scheme may add additional information such as context updates if required – however, the optional information can only be processed if broker and client currently use the same filter scheme. By relying on a unified, attribute-based subscription model, the broker can always fall back to default, attribute-based matching without taking any of the additional location information into account. This is especially important during transitions between filter schemes, as clients might receive some publications that were created with a different filter scheme before the transition is completed. This effect is further discussed in Section IV-C.

B. Transitions Between Filter schemes

To actually execute transitions between filter schemes, each filter scheme has to implement basic lifecycle methods as defined in [7]. These lifecycle methods are invoked during each transition, thereby ensuring a correct handover from one filter scheme to another. We define a transition $\mathcal{T}^{c}_{\mathrm{A}\to\mathrm{B}}$ as a unidirectional transformation function of all state associated to a client c from filter scheme A to filter scheme B.

The transition requires two subsequent steps: *state transfer* and *bootstrap*. *State transfer* does not involve any network communication, it solely operates on locally available information. Notifications are queued until the state transfer is finished, ensuring that each notification is processed exactly once. Thereby, the atomicity of the state transfer is ensured. Once all state is transferred successfully, filter scheme A is deactivated via the lifecycle management and B is activated. Since we cannot guarantee that all required state for B can be derived from information that is locally available at filter scheme A, the *bootstrap* phase may involve additional communication with the broker to obtain missing information. If state transfer between two filter schemes is not possible at all, the broker simply deletes the client's subscriptions during the state transfer phase and clients re-subscribe during the bootstrap phase utilizing the new filter scheme. This fallback ensures correct operation after the transition, but cannot guarantee delivery of notifications that are processed on the broker before the client's re-subscription arrived. Additionally, this approach can lead to high traffic overhead, if a large number of clients re-subscribes simultaneously.

On a mobile client, only one filter scheme can be active at a time, while brokers can optionally run multiple filter schemes in parallel to cater for several different areas or groups of clients. In our current prototype (c.f. Section III-C), the broker globally switches the filter scheme by executing $\mathcal{T}^{c}_{\mathrm{A}\to\mathrm{B}}$ for all its subscribers. This is motivated by the fact that broker networks already perform clustering of clients based on interest and – especially for location-based applications – based on their locations. As briefly discussed in Section II, we expect broker networks to become even more geographically distributed in the future. However, switching filter schemes on a broker for a subset of clients only could assist in dealing with heterogeneous devices or frequently changing network conditions and is subject of ongoing research. Most notably, parallel operation of multiple filter schemes as well as seamless switching between filter schemes is only possible with a common data model for notifications and subscriptions, as presented in the previous section. Thereby, incoming notifications that originate from a filter scheme other than the currently active one can still be processed at the broker and by clients.

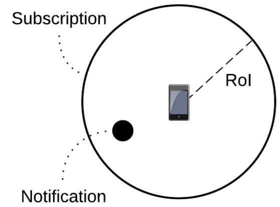
Transitions are initiated by the broker: to this end, it sends a control message to all affected clients and locally executes the respective transition in order to transfer the client's state to the new filter scheme. The control message contains the identifier of the new filter scheme and an optional set of parameters that are to be passed to the transition instance upon execution by the client. The set of parameters depends on the respective filter scheme and could, for example, contain an initial assignment of a client to a cell within a grid-based filter scheme, as discussed in the following section. We assume that control messages are delivered to clients via a reliable communication protocol. Consequently, all affected clients execute the transition locally. To reduce the overall complexity and coordination overhead of our system, we do not rely on any transactional scheme to notify the broker of finished transitions, leading to a brief period of time during a transition where clients and brokers run different filter schemes. We evaluate the impact of our proposed state transfer mechanims during this phase in Section IV-C.

C. Prototype and Realized Filter Schemes

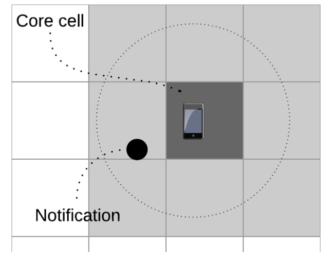
We realize the proposed design as an extension to an event dissemination system for location-based mobile services [18]. Within the prototype, we implement three representative filter schemes as presented in the following. Other relevant filter schemes within the scope of a location-based mobile application are discussed in Section V.

1) Location-based Filter Scheme: The location-based publish/subscribe filter scheme (termed LPS) utilizes placeholder variables to allow for complex context-based matching rules. Following the approach presented by Eugster et al. [9], a location-based subscription contains a placeholder variable for a mobile user's current position and a radius of interest. The actual value of the variable, i.e., the position of the mobile user, needs to be available at the broker. Therefore, clients need to send updated information to the broker whenever their location changes. Depending on the application at hand, these updates might be less frequent (e.g., for services like bus departure times at nearby stops) or very frequent (e.g., for an interactive augmented reality game).

The matching process involves calculating the intersections of the location associated to a notification with the area stated in the subscription, as illustrated in Figure 3a. Within this work, we perform matching on a circular area of interest defined by the *radius of interest* (RoI) around the current location of a mobile client. LPS can easily be extended in future work by including the movement speed and the direction of movement in the calculation of an area of interest as proposed in [2]. Compared to channel-based filter schemes where matching is based solely on hash comparisons, this consumes significantly more computational resources. The time required for the matching process can be reduced by matching subscriptions concurrently, as they are independent of each other. However, this still requires horizontal scaling, as the number of subscriptions is directly proportional to the number of active users in the system. As a consequence, the resources required at the broker can vary significantly based on the current load.



(a) Location-based filter scheme



(b) Grid-based filter scheme

Fig. 3. Within the LPS filter scheme, clients subscribe to a circular area around their current physical location. In GRID, clients subscribe to the cell that covers their current location (dark gray). In RGRID, clients additionally subscribe to all cells that are within the region of interest (light gray).

2) Grid-based Filter Schemes: Grid-based filter schemes rely on simple, channel-based filtering which reduces the complexity of the matching process as brokers no longer need to maintain a client's contextual information. To realize a location-based application, channels need to be mapped to locations (or regions). The mapping can either be done centrally at the broker or at the client itself using a well-known mapping function. Similar to [10], we focus on broker-based channel assignment, as this enables to update the grid assignment process based on load or other observed factors, such as client movement or exogenous events that are known in advance. Each cell of the grid corresponds to a channel in the publish/subscribe system.

In order to assign clients to grid cells, the following procedure is implemented: First, a client sends its current location to the broker, where it is mapped to a cell of the grid. The respective cell's geographic boundaries are then sent back to the client. The client checks whether he is about to leave the current cell and notifies the broker accordingly to retrieve the boundaries of a new cell, if necessary. If the broker alters the grid, for example as part of a load balancing process, the broker simply needs to send the resulting new cells to clients. This scheme greatly reduces the frequency of state updates issued by clients. Furthermore, fine-grained contextual data does not need to be stored and maintained by the broker, as the mapping function is stateless.

We implement two basic variants of a grid-based filter scheme, GRID and RGRID, based on the geocasting approach presented by Jodlauk et al. [10]. Within GRID, each client is assigned to a single cell. Consequently, the radius of interest stated by clients is not utilized by the broker. If a client is located near the border of a cell, it might not receive all relevant events, as illustrated in Figure 3b. To include the radius of interest, we implement a second version of the grid-based filter scheme, termed RGRID. In RGRID, the radius of interest stated by a client is used to determine the set of cells a client needs to subscribe to in order to cover the whole area of interest (c.f. Figure 3b). Compared to GRID, clients using RGRID simply subscribe to multiple channels to receive events within the whole radius of interest as defined by the application. Events are published to the channel that is associated to the client's current location. This ensures correct delivery of all relevant events at the cost of an increased number of irrelevant events (false positives) being sent to a subscriber. From a network provider's perspective, grid-based filter schemes could utilize a multicast-based message dissemination for a limited geographical region, as discussed in [10], thereby increasing their network efficiency.

3) State Transfer during Transitions: As discussed in Section III-B, state information needs to be transferred to enable seamless transitions between different filter schemes. For the filter schemes implemented in our prototype, the resulting transitions along with the respective state transfers are shown in Figure 4.

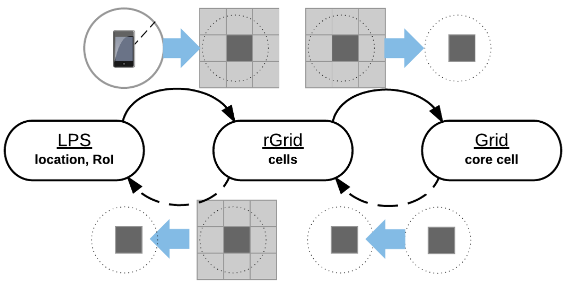


Fig. 4. Transitions and their associated state transformations. From LPS to RGRID, the position and radius of interest is used to determine the set of grid cells, the transition to GRID simply utilizes the location of the core cell. During transitions back to RGRID and LPS, state information such as the radius of interest and the exact location of a device is missing.

Within our prototype, client subscriptions are always transferred from one filter scheme to the other. Otherwise clients would frequently have to re-subscribe, leading to significant overhead and potential information loss during a transition. However, as illustrated in Figure 4, additional information such as the last known location of a client and its radius of interest is only available during the transition from the location-based filter scheme (LPS) to one of the grid-based filter schemes (GRID, RGRID). Using this information, the broker can directly assign clients to grid cells, without the need for additional status updates. During transitions in-between grid-based filter schemes, information on the current cell that a client is subscribed to is transferred. Information on a client's position or its radius of interest are not available at the broker. When switching from GRID to RGRID, the client detects whether it is necessary to send a location update (including the radius of interest) to retrieve missing grid cells.

While transitions from LPS to any of the grid-based filter schemes do not lead to loss of information at the broker, transitions back to the LPS filter scheme require status updates issued by the clients to restore state such as the last known position or the current radius of interest. This, in turn, can have a significant impact on the observed traffic characteristics during and right after a transition, as further discussed in Section IV-B. We evaluate the characteristics of our transition-enabled system in the following section, with a focus on the effects of state transfer during a transition.

IV. EVALUATION

We conduct an evaluation of our system prototype within the SIMONSTRATOR simulation environment [19]. The goals of the evaluation are twofold: (i) characterize the overall behavior of the transition-enabled system compared to static configurations (c.f. Section IV-B) and (ii) understand the impact of the provided state transfer mechanism on the performance of the system during and right after a transition (c.f. Section IV-C).

A. Simulation Scenario and Metrics

We model the workload of a multiplayer augmented reality game [17], as introduced in Section II. Client movement is based on OpenStreetMap data of the inner city of Darmstadt and the attraction points (portals) of the augmented reality game Ingress[1]. A part of the simulated area is shown in Figure 5, with nodes following walkways and grouping around specific points of interest. Clients move along pedestrian walkways towards attraction points, stay within their vicinity for one to five minutes, and start moving towards the next attraction point. The movement model determines the frequency of context updates required by a filter scheme. If a client stays within the vicinity of an attraction point, no context updates are required. Furthermore, the density of clients around an attraction point determines the fan-out of a notification. Every client subscribes to a circular area with 150 m radius around its current position and publishes one event per second containing 128 bytes payload (c.f. Table I).

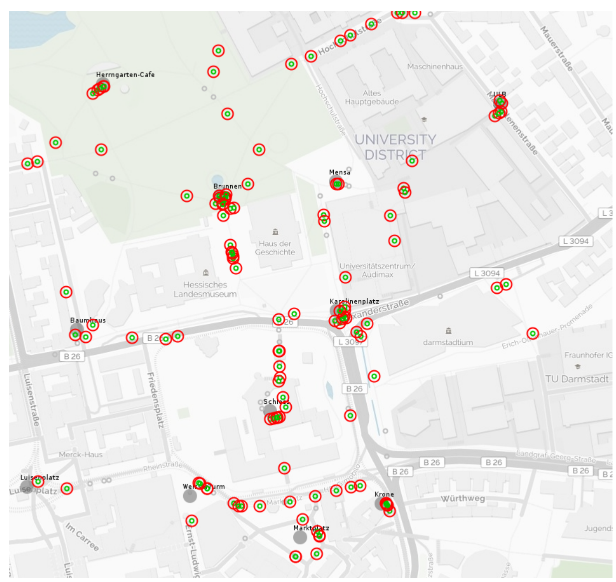


Fig. 5. Part of the simulated area within the central city of Darmstadt. Nodes move along pedestrian walkways and towards attraction points that are placed at specific locations on the map.

To assess the impact of transitions on the performance of the overall system, the following metrics are captured during the simulations.

[1]Available online: www.ingress.com/intel (a game account is required)

TABLE I
SCENARIO PARAMETERS

Simulated Area	$1.3\,\text{km} \times 1.3\,\text{km}$
Simulated Time	30 minutes
Cellular Network	Reliable, $200\,\text{ms}$ latency $\pm 100\,\text{ms}$
Pause Times	$1 - 5\,\text{min}$
Movement Speed	Pedestrian, $0.5 - 1.5\,\text{m/s}$
Radius of Interest	Circular, $r = 150\,\text{m}$
Event Generation	1 event per second per node, 128 byte payload

a) Recall: We measure the ratio between notified subscribed clients (true positives) and the total number of subscribed clients for a notification. If all subscribed clients were correctly notified, the recall of the system is 1.

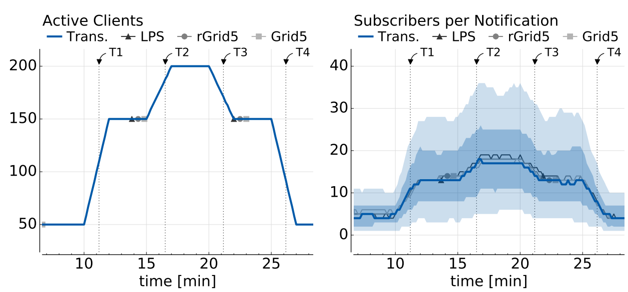
b) Precision: We assess the precision for each notification, as especially the grid-based filter schemes often lead to notifications being delivered to uninterested clients. The precision is defined as the ratio between notified subscribed clients and all notified clients. A precision of 1 indicates that the filter scheme never notifies uninterested clients.

c) Filter Complexity: As a measure for the computational complexity at the broker, we assess the number of match operations that are executed at the broker as the number of subscriptions an incoming notification has to be matched against. We do not distinguish between different match operators (e.g., hash comparison vs. area intersection).

d) Traffic and Overhead: The traffic at the broker and at the mobile clients is used as a measure for the overall load on the system and especially on the network infrastructure. We distinguish between payload (notifications and subscriptions) and additional overhead introduced by the filter schemes.

B. Transitions between Filter Schemes

We vary the number of active clients in the system over time according to Figure 6a. As each client subscribes to a circular area with $r = 150\,\text{m}$ around its current location, the number of clients that are notified of an event increases accordingly (c.f. Figure 6b). All plots show the median (solid line), the 25th and 75th percentile (dark area), and the 5th and 95th percentile (light area) over time for ten repetitions with different random seeds and a bin size of ten seconds.



(a) Active Clients (b) Notification Fan-Out

Fig. 6. Scenario with varying number of active clients and resulting fan-out of notifications as a consequence of increased node density. The markers (T1-T4) indicate the points in time where transitions are executed.

Transitions between individual filter schemes are triggered based on the number of subscribers at the broker, as shown in Figure 7. The resulting transitions are indicated by dotted vertical lines and the corresponding markers (T1 – T4) in all plots. While we do not focus on the derivation of meaningful rules for executing transitions, future work could include a machine learning approach to derive a set of suitable transition rules, as proposed in [6]. For GRID and RGRID, the simulated area is divided into a 5×5 grid, leading to cells with $260\,\text{m}$ edge length.

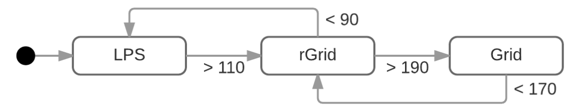
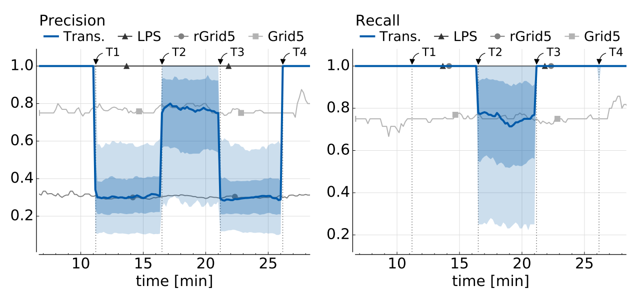


Fig. 7. Transitions between LPS, RGRID, and GRID are triggered by the broker based on the number of subscribed clients.

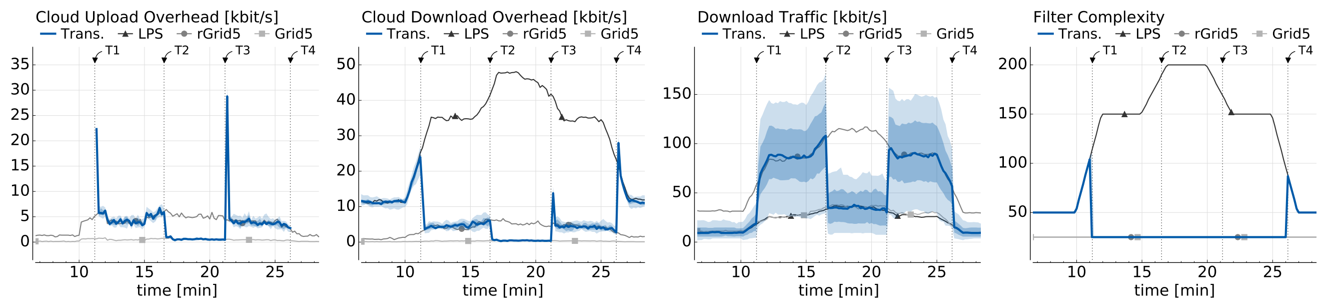
The performance characteristics of the transition-enabled system are shown in Figure 8. For comparison, the average performance of the individual filter schemes without transitions is included in the plots. The transitions and their impact on the performance are clearly visible and the overall system accurately reflects the characteristics of the individual filter schemes. When the system switches from LPS to RGRID (T1), the precision drops from 1.0 to between 0.2 and 0.4 for 50% of the clients (c.f. Figure 8a). This is due to the fact that clients subscribe to all cells that intersect with their current area of interest, leading to a high degree of irrelevant notifications. However, as the whole area of interest is still covered, the recall remains at 1.0 for all clients (c.f. Figure 8b). Notably, the recall is not affected by the transition T1, as our state transfer mechanism ensures correct transformation of location-based subscriptions to cell-based subscriptions on the broker. Without the state transfer mechanism, the performance is degraded during a transition, as discussed in Section IV-C.



(a) Precision (b) Recall

Fig. 8. Performance of the transition-enabled system in terms of precision and recall of notification delivery. The system adapts to the performance characteristics of the individual scheme right after a transition is executed.

When switching to GRID (T2), the recall drops significantly, while the precision again increases. Again, the characteristics of the GRID filter scheme are correctly reflected in the transition-enabled system: as clients subscribe only to a single cell regardless of the size of their actual region of interest, the position of a client within the cell determines the achievable



(a) Outgoing Control Traffic (b) Incoming Control Traffic (c) Clients' Download Traffic (d) Filter Complexity

Fig. 9. Overhead at the broker in terms of sent and received control messages increases during transitions as a consequence of cell updates sent to clients (T1 and T3) or missing location and radius of interest information sent by clients (T3 and T4). The download traffic observed by a single client and the filter complexity at the broker adapts to the respective filter scheme.

recall and precision. As the GRID filter scheme does not maintain the size of the area of interest at the broker, the transition back to RGRID (T3) leads to a slight degradation in the recall right after the transition. After missing state information such as the current grid cell identifier has been distributed to clients, the recall returns to one.

The process of state updates in the *boostrap* phase of a transition can be observed from the outgoing control traffic at the broker, shown in Figure 9a. For LPS, the broker does not distribute any state updates at all, while for GRID and RGRID, clients are notified of their corresponding grid cells when they move towards a new cell. Additionally, during the transition from LPS to RGRID (T1) and from GRID back to RGRID (T3), each client needs to receive an updated grid cell, leading to sharp increases in outgoing traffic. The same trend can be observed for the incoming control traffic at the broker (c.f. Figure 9b). While the traffic is decreased for the grid-based schemes compared to LPS due to less frequent client status updates, transitions lead to a sudden increase if missing state is requested by the broker. This is the case for the transition from GRID to RGRID (T3), where the radius of interest needs to be reported by clients, and for the transition from RGRID back to LPS (T4), where the client positions need to be updated.

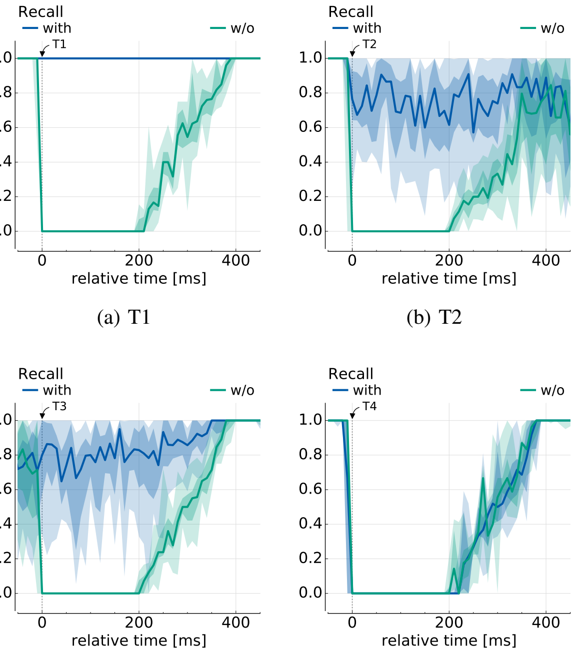
From a client's point of view, the upload traffic is not affected significantly, as each filter scheme requires all events to be uploaded to the broker. However, the precision of a filter scheme directly determines the resulting traffic for the mobile client, as shown in Figure 9c. While the grid-based filter schemes lead to increased download traffic for clients, they could be realized as multicast service, thereby increasing the overall network efficiency from a provider's view [10]. Regarding the complexity of matching operations (c.f. Figure 9d), both grid-based filter schemes match incoming notifications against the number of grid cells (25 in our scenario), leading to a fixed complexity independent of the number of clients. In LPS, the broker has to maintain one subscription per client due to the individual location information. Therefore, matching has to be performed separately for each subscription. As each client subscribes exactly once, the resulting filter complexity of LPS is equal to the number of clients in the system. However, we do not distinguish between match operations, which differ significantly between the considered schemes. Matching based on a simple hash (e.g., an identifier of a grid cell) is considered more efficient than calculating a lot of circular intersections as done by the LPS scheme. As optimized data structures might reduce this difference [8], we intentionally only consider the number of matching operations.

C. *Impact of the State Transfer Mechanism*

To assess the impact of the state transfer mechanism, we evaluated the recall during and after a transition with and without state transfer. Without state transfer, clients just re-subscribe using the new scheme as discussed in Section III-B. The observed recall during and shortly after the transitions T1-T4 is shown in Figure 10. As discussed in the previous section, transitions T1, T2, and T3 benefit from state transfer, maintaining a constant recall during the transitions. It takes one round trip time between clients and broker for the recall to recover if the state transfer mechanism is disabled, as clients need to be notified of the transition and issue a new subscription afterwards.

If no state information is transferred like during the transition from RGRID back to LPS (T4), the recall drops until the new subscriptions arrive at the broker (c.f. Figure 10d). This effect could be counteracted by a brief period of parallel operation of both filter schemes at the broker as proposed in [7], or by delaying notifications at the broker when a transition takes place.

The transition itself takes roughly 400 ms, as the communication channel between the coordinating broker and the respective mobile clients is assumed to be reliable with latency of 200 ms on average. It is a reasonable assumption that mobile clients can be reached via a reliable channel to trigger the transition, as packet drop over cellular mobile networks is very low due to lower-layer error correction schemes and centralized scheduling performed by base stations. However, the timing characteristics can be worse than assumed in our evaluation: more specifically, if the delay differs significantly in-between clients, there is a brief phase in which notification will not be delivered correctly to all interested clients. However, this is only the case if state



(a) T1

(b) T2

(c) T3

(d) T4

Fig. 10. Recall of the system with and without (w/o) state transfer mechanism during and after transitions T1-T4. No state information can be transferred during T4, causing a drop in the achieved recall for both configurations.

transfer form one scheme to the other is not possible, as is the case for transition T4. As long as the broker is able to correctly filter notifications as a consequence of the state transfer mechanism, all notifications will be delivered. By relying on a common, attribute-based model for notifications and subscriptions (as introduced in Section III-A), clients are still able to process notifications that originated from a different filter scheme than the currently active one.

As shown in this section, transitions between different filter schemes are a viable way to adapt a location-based publish/subscribe system to varying and dynamic conditions. Furthermore, by providing generic means to transfer meaningful state information between different filter schemes during a transition, the transition itself does not degrade performance.

V. Related Work

This section discusses relevant related works with respect to two main areas: (i) filter schemes for location-based publish/-subscribe and (ii) the adaptation of communication middleware to dynamics of mobile clients.

Numerous filter schemes for location-based subscriptions have been proposed in recent years [2], [10], [12], [23]. Leontiadis et al. [12] focus on vehicular movement, utilizing route data from a navigation system to determine and filter relevant events. Wang et al. [23] extend this concept with a generic filter concept, where subscriptions are defined for a given road segment and the direction of movement. Both filter schemes are targeted at vehicular movement speeds and rely on external data (e.g., map data) to determine the subscription area. We believe that our design can be adapted to vehicular scenarios as well, either by directly integrating the aforementioned filter schemes or by extending the cell structure and assignment process within the grid-based filter scheme (e.g., forming cells based on map data), as proposed in [10]. In future works, meaningful cells could be derived in an automated and adaptive fashion (e.g., based on current load and time of day) and clients could be updated accordingly by executing the corresponding transitions, without losing any of the generality of our approach.

Brimicombe et al. [2] present an extension of LPS [9], where the direction and speed of a client's movement is included in the subscription and in context updates to define a conic area of interest. Depending on the reported movement speed, the area of interest is enlarged in the direction of the movement. This is to counteract outdated state information about a client's position when filtering at the broker, which becomes especially important at higher movement speeds. We are currently extending our prototype to also take this information into account.

In [13], the authors propose a set of filtering algorithms for fixed broker networks that can be adapted during runtime to alter the tradeoff between network resource utilization (i.e., messaging overhead) and computational complexity. In contrast to our work, the filter algorithms do not support contextual information and it remains unclear whether such frequent changes can be handled efficiently by the proposed algorithms. However, an adaptive filter scheme could still benefit the broker network and would complement our work, considering that we are targeting the protocol between clients and their associated brokers.

Cao et al. [3] propose a publish/subscribe system where brokers are arranged in a two-level hierarchical topology, leading to significant savings in terms of network overhead and filter complexity. Although their system is limited to broker networks, our design exhibits similar characteristics: clients and their associated broker form the lower level of the hierarchy, while the broker network itself can be considered the upper level. Similar to [3], we strive to reduce complexity within the broker network while offering a fully-fledged location-based publish/subscribe for clients.

Chen et al. [4] propose an event-based system specifically targeted towards location-based services. In order to avoid the necessity of frequent location updates, events are only filtered based on coarse location data within the broker network. Fine-grained filtering is then performed on the mobile device itself, again trading complexity at the broker against network resource utilization and overhead for mobile clients. While the approach presented in [4] is rather static, it resembles the grid-based filter schemes presented in our work.

Ottenwälder et al. [15] propose a system for mobile complex event processing, where events are processed by a set of operators that are adapted to the mobile client's current location. Adaptation involves reconfiguring the operator graph as a consequence of client movement and resource availability.

Operators can be limited to a given spatial area, similar to location-based subscriptions discussed in our work. However, the work proposed in [15] focuses on the organization of operator graphs within the network of brokers, while our work explicitly covers the *last mile* between mobile clients and their associated broker. Bringing both concepts together is a promising direction for future work.

VI. Conclusion

In this paper, we present a method to execute transitions between different filter schemes within a location-based publish/subscribe system, allowing the system to adapt to the dynamic nature of mobile location-based applications. We show the applicability of our proposed method by integrating the respective functionality into an existing publish/subscribe system and evaluating the impact of transitions between filter schemes within the setting of a mobile augmented reality game. We highlight the impact of our state transfer mechanism that ensures seamless operation of the overall system while transitions are being executed. Our evaluation shows that transitions between filter schemes enable the overall system to adapt its performance and cost characteristics during runtime.

We believe that the proposed method can easily be extended to be applicable to a wide range of interesting use cases. One direction for future work is the integration of network mechanisms like cellular broadcast services to improve the network efficiency of grid-based filter schemes, similar to the mechanism proposed in [20]. Additionally, more elaborate filter schemes for vehicular mobility (e.g., street- and direction-based channel assignment) could be included to support a wide range of location-based automotive applications [12]. Lastly, transitions between filter schemes could be a valuable mean to increase location privacy and context confidentiality within publish/subscribe systems [14] based on user preferences: a user could explicitly switch to a less accurate filter scheme in terms of maintained contextual data to trade local resources against increased personal location privacy.

Acknowledgment

This work has been funded by the German Research Foundation (DFG) as part of projects C2, C3, and B1 within the Collaborative Research Center (CRC) 1053 – MAKI.

References

[1] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer, "Falling asleep with angry birds, facebook and kindle: a large scale study on mobile application usage," in *Proc. ACM MobileHCI*, 2011.

[2] A. Brimicombe and Y. Li, "Mobile space-time envelopes for location-based services," *Transactions in GIS*, vol. 10, no. 1, pp. 5–23, 2006.

[3] F. Cao and J. P. Singh, "Efficient event routing in content-based publish-subscribe service networks," in *Proc. IEEE INFOCOM*, 2004.

[4] X. Chen, Y. Chen, and F. Rao, "An efficient spatial publish/subscribe system for intelligent location-based services," in *Proceedings of the 2nd International Workshop on Distributed Event-based Systems (DEBS)*, 2003, pp. 1–6.

[5] L. Fiege, F. C. Gärtner, O. Kasten, and A. Zeidler, "Supporting mobility in content-based publish/subscribe middleware," in *Proc. ACM/IFIP/USENIX Middleware*. Springer-Verlag New York, Inc., 2003.

[6] A. Frommgen, R. Rehner, M. Lehn, and A. Buchmann, "Fossa: Learning eca rules for adaptive distributed systems," in *IEEE Int. Conf. on Autonomic Computing (ICAC)*. IEEE, 2015, pp. 207–210.

[7] A. Frömmgen, B. Richerzhagen, J. Rückert, D. Hausheer, R. Steinmetz, and A. Buchmann, "Towards the Description and Execution of Transitions in Networked Systems," in *Intelligent Mechanisms for Network Configuration and Security*. Springer, 2015, pp. 17–29.

[8] S. Ilarri, E. Mena, and A. Illarramendi, "Location-dependent query processing: Where we are and where we are heading," *ACM Computing Surveys (CSUR)*, vol. 42, no. 3, p. 12, 2010.

[9] K. Jayaram, P. Eugster, and C. Jayalath, "Parametric content-based publish/subscribe," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 2, p. 4, 2013.

[10] G. Jodlauk, R. Rembarz, and Z. Xu, "An optimized grid-based geocasting method for cellular mobile networks," in *18th ITS World Congress*, 2011.

[11] C. Kloch, E. B. Petersen, and O. B. Madsen, "Cloud based infrastructure, the new business possibilities and barriers," *Wireless Personal Communications*, vol. 58, no. 1, pp. 17–30, 2011.

[12] I. Leontiadis, P. Costa, and C. Mascolo, "A hybrid approach for content-based publish/subscribe in vehicular networks," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 697–713, 2009.

[13] G. Mühl, L. Fiege, and A. Buchmann, "Filter similarities in content-based publish/subscribe systems," in *Trends in Network and Pervasive Computing—ARCS 2002*. Springer, 2002, pp. 224–238.

[14] E. Onica, P. Felber, H. Mercier, and E. Rivière, "Confidentiality-preserving publish/subscribe: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 27, 2016.

[15] B. Ottenwälder, B. Koldehofe, K. Rothermel, K. Hong, D. Lillethun, and U. Ramachandran, "Mcep: a mobility-aware complex event processing system," *ACM Transactions on Internet Technology (TOIT)*, vol. 14, no. 1, p. 6, 2014.

[16] P. Pietzuch, D. Eyers, S. Kounev, and B. Shand, "Towards a Common API for Publish/Subscribe," in *Proc. ACM DEBS*, 2007, pp. 152–157.

[17] B. Richerzhagen, M. Schiller, M. Lehn, D. Lapiner, and R. Steinmetz, "Transition-enabled event dissemination for pervasive mobile multi-player games," in *Proc. 16th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2015.

[18] B. Richerzhagen, D. Stingl, R. Hans, C. Gross, and R. Steinmetz, "Bypassing the cloud: Peer-assisted event dissemination for augmented reality games," in *IEEE Conference on Peer-to-Peer Computing (P2P)*, 2014.

[19] B. Richerzhagen, D. Stingl, J. Rückert, and R. Steinmetz, "Simonstrator: Simulation and Prototyping Platform for Distributed Mobile Applications," in *Proc. EAI Conf. on Simulation Tools and Techniques (SIMUTOOLS)*. ACM, 2015.

[20] J. Rückert, J. Blendin, and D. Hausheer, "Software-defined multicast for over-the-top and overlay-based live streaming in isp networks," *Journal of Network and Systems Management*, vol. 23, no. 2, pp. 280–308, 2015.

[21] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang, "A first look at cellular network performance during crowded events," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1. ACM, 2013, pp. 17–28.

[22] P. Triantafillou and A. Economides, "Subscription summarization: A new paradigm for efficient publish/subscribe systems," in *Distributed Computing Systems, 2004. Proceedings. 24th International Conference on*. IEEE, 2004, pp. 562–571.

[23] H. Wang, T. Zhu, S. Lei, W. Yang, and Y. Wang, "Cmtg: a content-based mobile tendency geocast routing protocol in urban vehicular networks," *International Journal of Distributed Sensor Networks*, vol. 2015, p. 1, 2015.

[24] Y.-M. Wang, L. Qiu, D. Achlioptas, G. Das, P. Larson, and H. J. Wang, "Subscription partitioning and routing in content-based publish/-subscribe systems," in *16th International Symposium on DIStributed Computing (DISC'02)*. Citeseer, 2002.

[25] W. Wei, X. Zhu, and Q. Li, "Lbsnsim: Analyzing and modeling location-based social networks," in *Proc. IEEE INFOCOM*, 2014, pp. 1680–1688.

[26] M. P. Wittie, V. Pejovic, L. Deek, K. C. Almeroth, and B. Y. Zhao, "Exploiting locality of interest in online social networks," in *Proceedings of the ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. ACM, 2010, p. 25.

[27] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.