A Framework for Publish/Subscribe Protocol Transitions in Mobile Crowds

Björn Richerzhagen(✉), Alexander Wagener, Nils Richerzhagen, Rhaban Hark, and Ralf Steinmetz

Multimedia Communications Lab (KOM), Technische Universität Darmstadt, Darmstadt, Germany
{bjorn.richerzhagen,alexander.wagener,nils.richerzhagen, rhaban.hark,ralf.steinmetz}@kom.tu-darmstadt.de

Abstract. The increasing number of sensor-equipped mobile devices enables new applications for communication within crowds of people. Ranging from monitoring services that provide insights into the crowd's behavior to fully fledged messaging applications for end users, all such applications require communication protocols that are tailored to the characteristics of a mobile ad hoc network (MANET). A common communication scheme for such dynamic networks is the publish/subscribe (pub/sub) paradigm, that offers temporal and spatial decoupling. However, pub/sub protocols for MANETs are designed and evaluated under very restricted conditions regarding the expected mobility, number of users, application workload, and application requirements, targeting rather specific scenarios. In reality, such conditions are subject to change, especially considering people's behavior during crowded events. In this work, we propose a framework that enables dynamic transitions between different pub/sub protocols based on the currently monitored conditions or caused by an external trigger. We analyze the behavior during transitions and the overhead introduced by our framework through extensive simulations. By using simple pub/sub protocols and switching between them based on the observed conditions, we are able to maintain good service quality at reasonable overhead, thereby enabling applications to operate in dynamic conditions representative of mobile crowds.

1 Introduction

The trend towards mobile devices and their continuously increasing capabilities in terms of processing power, sensors, and communication interfaces motivates a range of applications for communication in crowds. Application objectives range from crowd sensing and monitoring, e.g., to detect critical densities of people, to messaging and infotainment [12,15]. Often, these applications are tailored towards a specific type of crowd (e.g., visitors of a music festival, or first responders [2,16]), where central infrastructure might be overloaded, too costly, or not available at all. In addition, user-generated content is often only relevant in vicinity of its creator [10], a property that can be exploited to increase system

R. Badonnel et al. (Eds.): AIMS 2016, LNCS 9701, pp. 16–29, 2016.
DOI: 10.1007/978-3-319-39814-3_2

performance and scalability [11,31]. Therefore, direct ad hoc communication is used to replace – or at least augment [23] – the communication via a central infrastructure. Consequently, a plethora of pub/sub systems for MANETs have been proposed in recent years, ranging from simple approaches based on notification or subscription flooding to more complex overlays that maintain distribution trees. Each approach focuses on a particular optimization target such as increasing robustness [19], providing low latency communication [20], or achieving a high resilience towards mobility [1]. Furthermore, each approach is able to deliver the desired performance only within bounded environmental conditions specific to the scenario it was initially designed for.

In this work we propose a framework that is able to dynamically exchange the utilized pub/sub protocol for a crowd of people, based on status information gathered within the system or based on extrinsic triggers. This allows us to adjust the performance vs. cost characteristics of the system based on the current conditions and application requirements, while at the same time relying on rather simple pub/sub protocols with well-known properties. Extending the concept of transitions and their corresponding lifecycles as proposed in [9], our framework addresses challenges related to the state transfer as well as message translation during the execution of a transition. We argue that existing protocols can be integrated easily, as they only need to provide access to stored subcriptions through a basic and easily extendable subscription model. By relaying network operations – e.g., sending and receiving of messages – as well as interaction with the application through simple proxy interfaces, transitions between protocols are executed transparent to the application.

Based on a simulative evaluation of a prototype of our proposed framework we quantify the performance benefits as well as the additional overhead introduced by executing transitions between two basic pub/sub protocols. The simulation setup is derived from real-world measurements of people's behavior during a large-scale music festival [8], thereby providing valuable insights into protocol performance in a realistic setting. Our findings reveal that transitions executed by our framework enable the system to adapt to changing conditions, while at the same time keeping the control overhead reasonably low. The system is able to maintain the desired performance characteristics and ensure delivery of notifications during and after the execution of a transition. The evaluation results further indicate the need for efficient bootstrapping mechanisms to improve the benefit of – especially short-lived – transitions. Overall, the execution of transitions between basic pub/sub protocols is a promising approach to tackle the challenge of communication in mobile crowds under dynamic conditions.

The remainder of this paper is structured as follows: In Sect. 2 we describe and discuss our proposed framework for transitions between pub/sub protocols in mobile crowds. We evaluate the framwork in a simulative setup derived from real-world crowdsourced measurements of people movement during a large-scale music festival and present the results in Sect. 3. After discussing relevant related work in Sect. 4, we conclude the paper and discuss future research directions in Sect. 5.

2 A Transition-Enabled Pub/Sub Framework

Figure 1 provides an overview of the entities of the transition-enabled pub/sub framework and their interactions. At the core of the framework, a number of pub/sub protocols are provided. Proper activation and termination of protocols during transitions is assured by the *lifecycle management*. Instead of directly interacting with the respective pub/sub protocol, applications publish, subscribe, and unsubscribe through an *interaction proxy*. To this end, the proxy takes care of relaying the respective method calls to the currently active pub/sub protocol. Additionally, it handles callbacks and notifications triggered by the pub/sub protocol upon reception of a notification and forwards the notification to the application for further processing. The methods provided by the proxy follow the proposal for a common pub/sub application programming interface, as defined in [21]. This ensures that transitions between protocols are transparent to the application – no modification of application code is required.

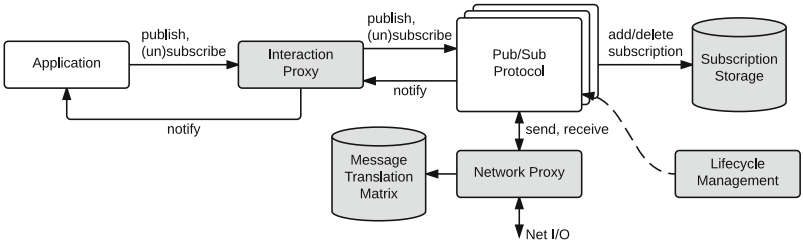


Fig. 1. Overview of the entities and their interactions in the proposed framework.

The currently active pub/sub protocol stores its subscriptions – local subscriptions originating from the application, as well as subscriptions managed on behalf of other nodes – in the provided *subscription storage*. To this end, a uniform abstraction for subscriptions is required. This abstraction is provided by the interaction proxy and further discussed in Sect. 2.2. To enable communication among nodes in the system, each pub/sub protocol sends and receives messages through a common *network proxy* provided by the framework. The network proxy relays incoming messages to the currently active pub/sub protocol. During protocol transitions, some nodes may receive messages originating from a different protocol than their currently active one. Rather than simply discarding these messages, the network proxy attempts to resolve basic information and translate it into a form that can be processed by the currently active pub/sub protocol. This (optional) functionality is implemented on a per-protocol basis by adding the respective transformation functions to the *message translation matrix*. In the following section, the process of a transition within our framework is discussed in detail.

2.1 Execution of Transitions

The process of a transition itself is shown in Fig. 2. For simplicity, we assume that there exists a logically centralized coordinator. For most scenarios, this assumption is reasonable, as applications usually report their data to some kind of sink infrastructure, e.g., the coordination unit of an emergency response team. In cases involving multiple coordinators, distributed consensus protocols [18] could be used to agree on a transition. The coordinator starts a transition in consequence to a trigger event. Such an event can be the outcome of monitored state (e.g., a sharp decrease in system performance, or sudden increase in people movement), or based on an external trigger. Based on the target pub/sub protocol, a transition strategy is selected and the decision is disseminated to all mobile nodes. Dissemination can either be performed by the pub/sub protocol itself (in this case, the framework registers as a subscriber to the respective event type) or via an out-of-band dissemination protocol.

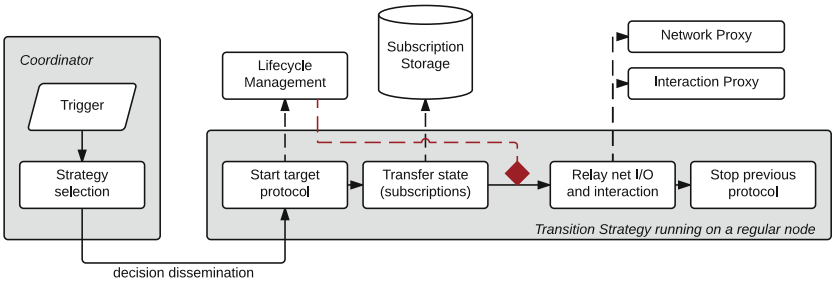


Fig. 2. Control flow and execution steps for a transition.

Once the decision arrives at a mobile node, the respective transition strategy is executed. Each strategy consists of four basic steps: (i) start the new protocol through the lifecycle manager, (ii) transfer state to the new protocol, relying on the subscription storage, (iii) relay application and network interaction to the new protocol by utilizing the respective proxies, and (iv) terminate the previously running protocol. Before actually relaying all interaction to the new protocol, the strategy needs to wait until the lifecycle manager signals successful startup of the new protocol, as discussed in [9]. Depending on the protocol, startup might involve the formation of a topology and, as a consequence thereof, some message exchange unrelated to the application payload. The network proxy allows such communication during the startup and shutdown phase of a protocol, thereby ensuring the functionality of bootstrap mechanisms and a graceful shutdown. Once the new protocol is ready, all further interaction is relayed through the new protocol and the old one is terminated.

Without any additional information about the pub/sub protocols involved in a transition, the framework is already able to execute a transition by only relying on the common abstraction of the subscription storage as well as the

interaction proxy. However, such a transition would involve unsubscribing with the old protocol and then re-subscribing for all locally stored subscriptions afterwards, introducing significant overhead. Therefore, state transfer can be refined on a per-protocol basis, leading to more evolved transition strategies. If, for example, the target protocol and the currently active protocol have a notion of brokers (e.g., some nodes in the mobile network act as message brokers serving other nodes that act solely as clients), one can bootstrap the new protocol by assigning the role of a broker to the nodes that already previously acted as brokers. Thereby, state does not need to be redistributed during startup, but instead only as a consequence of the default maintenance mechanisms of the new protocol. This strategy significantly reduces the overhead caused by a transition.

As our framework is used for communication within a crowd of people, we demand the whole network to execute transitions. In a MANET, partitions might occur and decisions might not reach all affected nodes, potentially leading to protocol misbehavior once partitions merge. To this end, the network proxy piggybacks an identifier of the currently active protocol, the transition strategy, as well as a simple logical clock to messages sent by a node. If another node receives such a message, but does not yet employ the same protocol, the appropriate transition is executed. In cases where the sending node employs an outdated protocol, the identifier helps in determining whether the message can be translated and still be processed by the new protocol. Translation of messages is enabled as all protocols deploy a common subscription model, which is discussed in the following section.

2.2 Subscription Model and Storage

As our framework utilizes a common storage for subscriptions, all pub/sub protocols need to use or extend our basic subscription model. To this end, we deploy an attribute-based subscription scheme that supports dynamic types, as illustrated in Fig. 3. In addition to an arbitrary number of types and attributes, each subscription may carry one string-typed attribute denoting a topic. In this way, the resulting subscriptions can be processed by attribute-based and simple channel- or topic-based pub/sub protocols. Subscriptions (and notifications) are built by the application, relying on factory methods provided by the interaction proxy. Therefore, applications do not depend on subscription schemes introduced by the individual protocols.

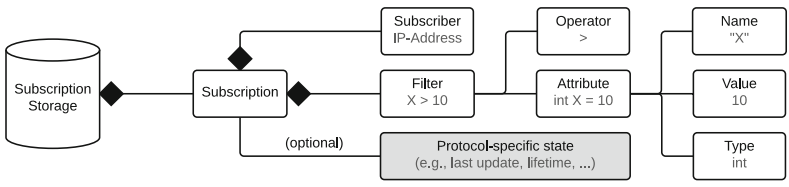


Fig. 3. Subscription model within the proposed framework.

To enable protocol-independent management of subscriptions, the respective objects that are used to compose a subscription (e.g., filters and attributes) are created through factory methods provided by the interaction proxy. Therefore, if an existing pub/sub protocol needs to extend the basic subscription model, it can simply extend the respective factory method to include additional, protocol-specific state within a subscription. The same model and factory concept is also used by applications for the creation of notifications. If nodes act as brokers within the current protocol (i.e., they manage subscriptions on behalf of other nodes), the subscribers are also attached to the respective subscription. In the basic model, only an IP address is required for each subscriber.

2.3 Integrating Existing Pub/Sub Protocols

One goal of our framework is to keep the required modifications of existing protocols as simple as possible. To ensure basic interoperabilit one needs to (i) provide an adapter for the interaction proxy and the network proxy, (ii) utilize the subscription storage, and (iii) implement the basic lifecycle methods. While the current network proxy requires some modifications (the use of a custom socket), a future version of the framework could intercept[1] calls to the respective methods. As a basic lifecycle handling is already part of most protocols, and our subscription model is fairly generic, (ii) and (iii) do not pose significant challenges and only require minor changes.

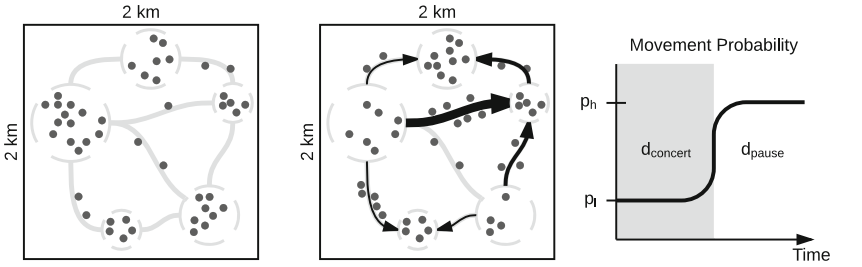
To enhance system performance during a transition, one may provide a custom transition strategy for the respective protocol. As discussed in Sect. 2.1, a strategy might include more complex state exchange mechanisms to reduce messaging overhead. Usually, the resulting state exchange mechanisms depend on specific combinations of source and target protocol. If a matching state exchange mechanism is provided for the current transition, it is executed automatically by our framework. In addition to that, protocols might also provide custom entries to the message translation matrix. The matrix simply consists of descriptions of messages used by a protocol and pointers to some common data fields within those messages, like subscriptions or notifications. In this way, incoming messages of the wrong protocol type can be interpreted by the currently running protocol to some extent.

We aim to support simple integration of existing pub/sub protocols into the transition-enabled framework by requiring only minor modifications. However, the framework allows protocol designers to provide more elaborate transition strategies tailored towards the peculiarities of the specific protocols.

3 Prototype and Evaluation

We implemented a Java-based prototype of the proposed framework on top of the Simonstrator-Platform [24] to enable both simulative, and prototypical performance studies. Goal of the evaluation is to verify the concept of transitions

[1] We are currently extending the prototype to rely on Java Reflections for this purpose.



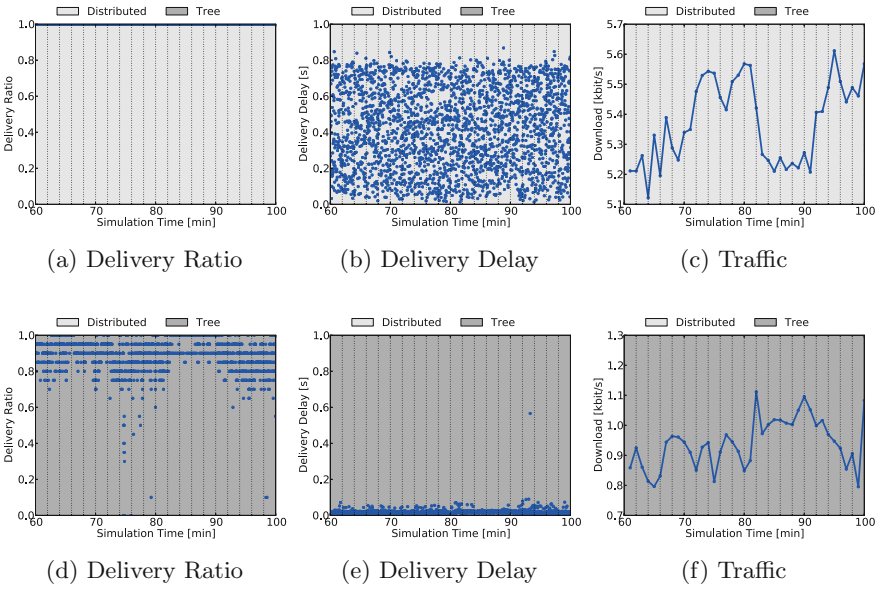
(a) During concerts. (b) During pauses. (c) Movement probability.

Fig. 4. Illustration of the evaluation scenario, derived from crowdsourced movement profiles of visitors of a music festival [8].

in pub/sub systems for crowd communication as a viable solution to react to environmental changes in a flexible and generic way. Therefore, we model a scaled-down scenario based on real-world crowdsourced measurements of people movement during a music festival [8], as illustrated in Fig. 4. The scenario can roughly be categorized into two phases with respect to movement characteristics: a relatively static phase during concerts (c.f. Fig. 4a), and an active phase where the crowd moves to different stages or enters and leaves the festival area (c.f. Fig. 4b). Stages and other relevant points of interest (e.g., food booths) are modeled as attraction areas, allowing us to rely on a modified version of the SLAW movement model [14,30]. In our evaluation scenario, we vary the probability that a node selects a new attraction area and starts moving towards it, mimicking real-world behavior during and after a concert (c.f. Fig. 4c).

The evaluation was conducted by means of simulation on the Simonstrator-Platform [24], relying on the overlay simulator PeerfactSim. KOM [29] and the NS-3 802.11 g underlay model [26]. We simulated 400 nodes on a $2000\,m \times 2000\,m$ area, with a publication rate of 1 publication per second and a fan-out of 1 : 20, meaing each publication had 20 interested subscribers. Two hours of operation are simulated: (i) during the first 30 min all nodes join; (ii) after 40 min, the workload phase starts and subscriptions, as well as publications, are issued; (iii) performance and cost metrics are measured between minute 60 and 100; (iv) the workload is applied until the end of the simulation. Each run was repeated five times with different random seeds, and the plots show the average over these five runs. The scenario starts with a concert phase lasting until minute 70. During this concert phase, the probability of a node to start moving to a new attraction area is set to $p_l = 0.05$. Two minutes before the end of the concert, this probability starts to increase linearly, until it reaches $p_h = 0.7$ at minute 72, leading to a noticeable number of nodes moving around in-between attraction areas. The probability is reduced back to $p_l = 0.05$ around minute 80, when the next concert starts. This cycle is repeated once more around minute 90.

For the evaluation, two basic pub/sub protocols are integrated into the framework: (i) an approach using notification flooding through a gossiping mechanism



(a) Delivery Ratio (b) Delivery Delay (c) Traffic

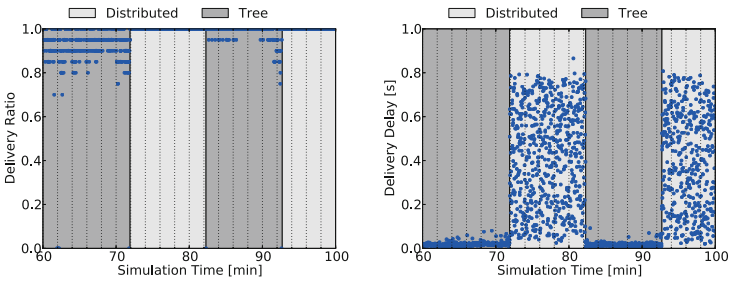
(d) Delivery Ratio (e) Delivery Delay (f) Traffic

Fig. 5. Performance in terms of delivery ratio and delay, and cost in terms of traffic for static deployment of the distributed (a–c) and the tree-based (d–f) pub/sub protocol in the dynamic scenario.

(termed *distributed* in the following), and (ii) a broker-based approach, where a subset of the nodes is elected as brokers that maintain subscriptions on behalf of the remaining nodes. In the broker-based approach, notifications are sent along a delivery tree formed by the broker nodes, thereby reducing the traffic. To understand the behavior of both protocols in our dynamic scenario, we evaluated them within our framework without triggering any transitions. The resulting performance and cost characteristics over time are shown in Fig. 5.

The gossip-based approach is robust enough to maintain a delivery ratio of one over the whole duration of the simulation (c.f. Fig. 5a). However, due to the utilization of a gossiping mechanism, event delivery can take up to one second (c.f. Fig. 5b), and the resulting traffic is five times higher than for the broker-based approach that forms a delivery tree (c.f. Fig. 5c and f). While delivery ratio and delay remain constant even during phases of high mobility, a slight increase in average traffic (c.f. Fig. 5c) can be observed. For the broker-based approach, the formation of delivery trees in the dynamic scenario leads to an average delivery ratio of 0.9 in the more static phases of the scenario (c.f. Fig. 5d). However, the ratio falls to an average 0.8 with significantly worse performance for a fraction of the nodes during phases with higher mobility. This is the result of failures in the delivery tree, caused by brokers that move out of range of each other. Until the failure is detected and repaired, the performance suffers. As long as the delivery tree remains functional, the broker-based approach achieves significantly better delivery delays (c.f. Fig. 5e) and consumes less traffic than the gossip-based approach.

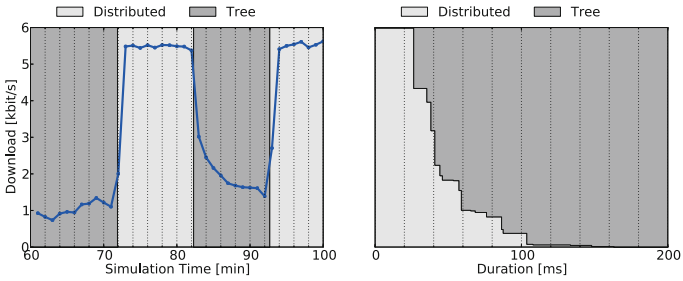
We conduct an evaluation featuring transitions between the broker-based and the gossip-based approach in order to provide a proof-of-concept evaluation of our proposed framework, as well as for an estimation of the cost introduced by the framework to control transitions. Transitions are triggered shortly after the user-mobility changes, according to the scenario description. The results for the transition-enabled framework are shown in Fig. 6. The background colors of the plots denote the currently active protocol. It can easily be seen that a transition is triggered a few minutes after the mobility starts to increase, as it becomes more probable for nodes to select a new attraction area. At minute 72, the transition from the broker-based approach to the distributed gossiping-based approach takes place. As a consequence, the delivery ratio directly increases, as the more robust protocol is used (c.f. Fig. 6a). However, this transition also leads to a significant increase in variation of the delivery delay (c.f. Fig. 6b) and an increased overall traffic consumption (c.f. Fig. 7a).



(a) Delivery Ratio (b) Delivery Delay

Fig. 6. Transitions between the broker-based and the distributed, gossip-based protocol occur at minutes 72, 82, and 92. The performance characteristics of the system change accordingly.

As the gossip-based protocol relies on notification flooding, nodes only need to filter based on their own, locally stored subscriptions. Therefore, there is no need for state transfer when initializing the protocol. This becomes apparent from the sharp transition of the traffic at minute 72, as shown in Fig. 7a. In contrast to that, the transition back to the broker-based approach at minute 82 requires some initial bootstrapping and, thus, causes higher traffic, as brokers have to be elected. Due to the scale of the x-axis, one might get the impression of a discrete switch between pub/sub protocols. A zoomed-in version of the first transition, shown in Fig. 7b, reveals that it takes up to 180 ms in our setting until the last node executes the transition. As shown in our evaluation, the framework successfully executes transitions between two different pub/sub protocols during runtime. The performance of the overall system remains stable during and after transitions. Our proposed approach, thus, is able to adapt to the dynamics of the considered scenario by switching between protocols, as initially motivated.



(a) Traffic (b) Decision distribution

Fig. 7. Bootstrapping the tree-based protocol consumes more traffic directly after the transition at minute 82. A zoomed-in version of the transition at minute 82 shows how the decision spreads in the network.

4 Related Work

There exists a number of reconfigurable and adaptive pub/sub middleware concepts for fixed [6,13] and mobile networks [27,28]. As our work specifically targets mobile crowds, we focus on the latter and additionally discuss related, self-adaptive pub/sub protocols.

Sivaharan et al. propose GREEN [27], a component-based middleware for Internet-scale pub/sub, inspired by REDS [6]. In contrast to REDS, the authors specifically aim at integrating local MANET regions into the overall pub/sub system. To this end, mobile devices using the middleware can be configured to act as gateways for nearby devices. The device-to-device communication is then handled via a range of configurable protocols that can be integrated into GREEN. However, the authors do not evaluate or discuss the implications of dynamic reconfigurations on mobile networks and solely consider a protocol based on notification-flooding, thereby obviating the need for any state transfer or maintenance mechanism. With our framework, the pub/sub middleware can be extended to actually react to environmental changes and dynamics by executing protocol transitions on the fly.

Sørensen et al. propose a middleware for context-aware applications on mobile networks [28]. While focusing explicitly on providing a higher level programming abstraction to pervasive applications, the proposed middleware also includes a communication substrate that provides pub/sub capabilities. The authors base their pub/sub substrate on the design of the MANET pub/sub protocol STEAM [17]. Unfortunately, no performance figures are given, and adaptability to changing dynamics is not evaluated. According to a recent survey of middleware approaches for pervasive applications by Raychoudhury et al. [22], event-based communication is a key building block for all kinds of middleware solutions. Therefore, we believe that our framework poses a valuable foundation for middleware concepts, such as [28]. Instead of proposing a new middleware concept as in [3,28], we aim at providing a core functionality as a potential

component of such a middleware in a lightweight fashion, putting a focus on reconfigurability. Therefore, by integrating our framework as communication substrate as envisioned by Sørensen et al. and providing higher-level pub/sub functionality on top, one can realize mobile applications that can acutally adapt their local ad hoc communication means to dynamic environmental conditions.

Depending on the characteristics of the application and the environmental conditions, more elaborate pub/sub protocols can be utilized and incorporated into our framework. For rather static environments but large amounts of nodes, hierarchical approaches, as presented by Yoo et al. [32], can be deployed. By relying on fixed delivery trees, and limiting communication to the exchange of subscriptions between dedicated broker nodes, the overhead caused by flooding of notifications can be avoided. However, such a structured approach suffers from node mobility. To this end, a semi-structured, density-based approach has been presented by Friedmann et al. [7]. Here, routing between brokers relies on a gradient-based scheme, making it more robust against broker movement. Additionally, the election of brokers requires only very limited knowledge – the density of the two-hop neighborhood – and, thus, can be performed without adding significant overhead. To address high mobility, or frequent connection failures, completely topology-less approaches have been proposed. Extending flooding-based concepts, approaches such as [19] utilize gossip-based communication protocols to limit the amount of redundant data transmissions.

Within our proposed framework, one can leverage the benefits of these different approaches by switching between pub/sub protocols at runtime and choosing the protocol that best fits the current conditions. At the same time, application developers just utilize a simple and well-known pub/sub interface, as all functionality of our proposed framework remains transparent to the application.

5 Discussion and Conclusion

A number of interesting application concepts rely on direct ad hoc communication in crowds of people, where cellular infrastructure might be unavailable or simply overloaded. The performance of the utilized communication protocol, however, can vary significantly depending on the dynamics of the considered scenario. In this work, we propose a framework that is able to switch between different pub/sub protocols for mobile networks at runtime. To this end, the framework provides a basic abstraction for subscriptions, as well as mechanisms for state transfer and interaction with the network and the application, allowing to easily integrate new protocols. Based on real-world measurements of a music festival, we derive a representative workload for a communication system in such an environment, and use it to evaluate a prototype of our proposed framework including two basic pub/sub protocols. In contrast to a static configuration of the protocols, our framework is able to maintain the desired performance vs. cost characteristics even in the face of dynamic changes of network conditions by executing transitions between the different protocols.

Our framework can work as a wrapper for existing pub/sub middleware, as the application interface relies on default pub/sub methods (c.f. [21]) defined by

the interaction proxy. To trigger transitions, the pub/sub system itself can be utilized: in such a case, the lifecycle management simply acts as a subscriber to a well-defined control channel. It is beyond the scope of this work to provide means to accurately monitor the network state as a foundation for transition execution – however, we are currently integrating an adaptive monitoring system that is targeted towards similar scenarios [25].

We are currently in the process of integrating a wider range of more complex pub/sub protocols into our framework, especially looking into protocols that support location-based or context-based publish/subscribe semantics. These protocols offer richer subscription semantics, enabling more sophisticated filtering of relevant notifications. One interesting application for the proposed framework is switching between direct delivery of notifications and delay-tolerant protocols [4,5]. Here, our framework could help in preserving energy by disabling the direct communication interfaces opportunistically, as consequence of a protocol transition, if applications do not require instant delivery for certain types of events. However, this would require centralized coordination via a cellular interface, as previously explored in [23]. The prototype of the framework is available online as part of the Simonstrator platform [24].

Acknowledgments. This work has been funded by the German Research Foundation (DFG) as part of projects B1, C2 and C3 within the Collaborative Research Centre (CRC) 1053 – MAKI. Ralf Steinmetz has been partially supported by a Chair of Excellence from University Carlos III, Madrid, Spain. The authors would like to thank Sophie Schönherr for her valuable contributions.

References

1. Anceaume, E., Datta, A.K., Gradinariu, M., Simon, G.: Publish/subscribe scheme for mobile networks. In: Proceedings of the POMC (2002)
2. Baldini, G., Karanasios, S., Allen, D., Vergari, F.: Survey of wireless communication technologies for public safety. IEEE Commun. Surv. Tutorials **16**(2), 961–987 (2014)
3. Batista, T.V., Joolia, A., Coulson, G.: Managing dynamic reconfiguration in component-based systems. In: Morrison, R., Oquendo, F. (eds.) EWSA 2005. LNCS, vol. 3527, pp. 1–17. Springer, Heidelberg (2005)
4. Boldrini, C., Conti, M., Passarella, A.: Design and performance evaluation of contentplace, a social-aware data dissemination system for opportunistic networks. Comput. Netw. **54**(4), 589–604 (2010)
5. Costa, P., Mascolo, C., Musolesi, M., Picco, G.P.: Socially-aware routing for publish-subscribe in delay-tolerant mobile ad hoc networks. IEEE J. Selected. Areas Commun. **26**(5), 748–760 (2008)
6. Cugola, G., Picco, G.P.: Reds: a reconfigurable dispatching system. In: Proceedings of the 6th International Workshop on Software Engineering and Middleware, pp. 9–16. ACM (2006)
7. Friedman, R., Kaplun Shulman, A.: A density-driven publish subscribe service for mobile ad-hoc networks. Ad Hoc Networks (2012)

8. Frömmgen, A., Heuschkel, J., Jahnke, P., Cuozzo, F., Schweizer, I., Eugster, P., Mühlhäuser, M., Buchmann, A.: Crowdsourcing measurements of mobile network performance and mobility during a large scale event. In: Karagiannis, T., et al. (eds.) PAM 2016. LNCS, vol. 9631, pp. 70–82. Springer, Heidelberg (2016). doi:10.1007/978-3-319-30505-9_6
9. Frömmgen, A., Richerzhagen, B., Rückert, J., Hausheer, D., Steinmetz, R., Buchmann, A.: Towards the description and execution of transitions in networked systems. In: Latré, S., Charalambides, M., François, J., Schmitt, C., Stiller, B. (eds.) AIMS 2015. LNCS, vol. 9122, pp. 17–29. Springer, Heidelberg (2015)
10. Huguenin, K., Kermarrec, A., Kloudas, K., Taïani, F.: Content and geographical locality in user-generated content sharing systems. In: Proceedings of the NOSSDAV (2012)
11. Jaho, E., Stavrakakis, I.: Joint interest-and locality-aware content dissemination in social networks. In: Proceedings of the WONS (2009)
12. Karaliopoulos, M., Telelis, O., Koutsopoulos, I.: User recruitment for mobile crowdsensing over opportunistic networks. In: Proceedings of the INFOCOM, pp. 2254–2262. IEEE (2015)
13. Leclercq, M., Quéma, V., Stefani, J.B.: Dream: a component framework for the construction of resource-aware, reconfigurable moms. In: Proceedings of the 3rd Workshop on Adaptive and Reflective Middleware, pp. 250–255. ACM (2004)
14. Lee, K., Hong, S., Kim, S.J., Rhee, I., Chong, S.: Slaw: A new mobility model for human walks. In: Proceedings of the INFOCOM 2009, pp. 855–863. IEEE (2009)
15. Ma, H., Zhao, D., Yuan, P.: Opportunities in mobile crowd sensing. IEEE Commun. Mag. **52**(8), 29–35 (2014)
16. Martín-Campillo, A., Crowcroft, J., Yoneki, E., Martí, R.: Evaluating opportunistic networks in disaster scenarios. J. Netw. Comput. Appl. **36**(2), 870–880 (2013)
17. Meier, R., Cahill, V.: STEAM: Event-based middleware for wireless ad hoc network. In: Proceedings of the DEBS, pp. 639–644, Jul 2002
18. Olfati-Saber, R., Fax, A., Murray, R.M.: Consensus and cooperation in networked multi-agent systems. Proc. IEEE **95**(1), 215–233 (2007)
19. Paridel, K., Vanrompay, Y., Berbers, Y.: Fadip: lightweight publish/subscribe for mobile ad hoc networks. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6427, pp. 798–810. Springer, Heidelberg (2010)
20. Pietzuch, P.R., Bacon, J.: Hermes: a distributed event-based middleware architecture. In: Proceedings of the DEBS (2002)
21. Pietzuch, P., Eyers, D., Kounev, S., Shand, B.: Towards a common API for Publish/Subscribe. In: Proceedings of the DEBS, pp. 152–157. ACM (2007)
22. Raychoudhury, V., Cao, J., Kumar, M., Zhang, D.: Middleware for pervasive computing: a survey. Pervasive Mobile Comput. **9**(2), 177–200 (2013)
23. Richerzhagen, B., Stingl, D., Hans, R., Gross, C., Steinmetz, R.: Bypassing the cloud: Peer-assisted event dissemination for augmented reality games. In: Proceedings of the P2P, pp. 1–10. IEEE (2014)
24. Richerzhagen, B., Stingl, D., Rückert, J., Steinmetz, R.: Simonstrator: Simulation and prototyping platform for distributed mobile applications. In: Proceedingd of the SIMUTOOLS. ICST (2015)
25. Richerzhagen, N., Stingl, D., Richerzhagen, B., Mauthe, A., Steinmetz, R.: Adaptive monitoring for mobile networks in challenging environments. In: Proceedings of the ICCCN, pp. 1–8. IEEE (2015)
26. Riley, G., Henderson, T.: The NS-3 network simulator. In: Wehrle, K., Güneş, M., Gross, J. (eds.) Modeling and Tools for Network Simulation. Springer, New York (2010)

27. Sivaharan, T., Blair, G.S., Coulson, G.: GREEN: a configurable and re-configurable publish-subscribe middleware for pervasive computing. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3760, pp. 732–749. Springer, Heidelberg (2005)
28. Sørensen, C.F., Wu, M., Sivaharan, T., Blair, G.S., Okanda, P., Friday, A., Duran-Limon, H.: A context-aware middleware for applications in mobile ad hoc environments. In: Proceedings of the MPAC, pp. 107–110. ACM (2004)
29. Stingl, D., Gross, C., Ruckert, J., et al.: PeerfactSim. KOM: a simulation framework for peer-to-peer systems. In: Proceedings of the HPCS. IEEE (2011)
30. Stingl, D., Richerzhagen, B., Zollner, F., Gross, C., Steinmetz, R.: Peerfactsim. kom: Take it back to the streets. In: Proceedings of the HPCS, pp. 80–86. IEEE (2013)
31. Vulimiri, A., Michel, O., Godfrey, P., Shenker, S.: More is less: reducing latency via redundancy. In: Proceedings of the HotNets (2012)
32. Yoo, S., Son, J., Kim, M.: A scalable pub/sub system for large mobile ad hoc networks. J. Syst. Softw. **82**, 1152–1162 (2009)