Flexub: Dynamic Subscriptions for Publish/Subscribe Systems in MANETs

Engineer Bainomugisha[1], Koosha Paridel[2], Jorge Vallejos[1], Yolande Berbers[2], and Wolfgang De Meuter[1]

[1] Software Languages Lab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Elsene, Brussels, Belgium
{ebainomu,jvallejo,wdmeuter}@vub.ac.be
[2] Department of Computer Science, K.U. Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium
{koosha.paridel,yolande.berbers}@cs.kuleuven.be

Abstract. Current publish/subscribe systems provide very limited support to modify subscriptions dynamically. Consequently, they cannot efficiently control the flow of events between publishers and subscribers, which may lead to unnecessary network traffic. In addition, it is not possible to automatically subscribe or unsubscribe to a service depending on certain context of use. This implies for developers to manually manage subscriptions (e.g., taking care of when to cancel or re-issue a subscription), which may result in inappropriate subscription states (e.g., subscriptions that are cancelled too late). In this paper, we propose the concept of *dynamic subscription mechanisms* that improves the expressiveness and flexibility of subscriptions. We introduce a new dimension to a subscription that allows a subscriber to express the flow of matched events, and when a new subscription can be (re)issued. We validate our claims for improved flexibility and expressiveness by providing language abstractions and a prototype implementation of a dynamic subscription mechanism framework called *Flexub* that supports a variation of subscription mechanisms. When compared to existing subscription models, our experiment results show that the support for *dynamic subscription mechanisms* greatly reduces network traffic of events sent from publishers to the subscribers. In addition, our approach reduces the workload on the subscriber side.

Keywords: Publish/Subscribe systems, subscription mechanisms, mobile ad hoc networks, context-aware systems.

1 Introduction

Event filtering techniques and subscription models are two important areas of research for Publish/Subscribe systems [10,3,6,9]. The works on event filtering and subscription models propose various ways for specifying the interests of a subscriber and deciding which events the subscriber should receive. However, emerging mobile applications for MANETs such as context-aware applications,

K.M. Göschka and S. Haridi (Eds.): DAIS 2012, LNCS 7272, pp. 132–139, 2012.

require more expressive subscription models, as it is desirable for subscribers to be able to dynamically modify subscriptions depending on some context conditions. For instance, a subscriber may need to specify that a subscription should be automatically adapted or cancelled after a certain number of matched events are received.

Current solutions suffer from limited expressiveness and provide insufficient support to modify subscriptions dynamically. While event filters [10] can be used to discard events that are of no interest to the subscriber, it is not possible to express when a subscription should be cancelled or modified. Consequently, subscribers have a limited control on the flow of matched events, which may lead to unnecessary network traffic. Moreover, it is not possible to automatically subscribe or unsubscribe to a service depending on certain context conditions (e.g., battery level, number of required events, location, etc.). This implies for developers to manually manage subscriptions (e.g., taking care of when to cancel or re-issue a subscription), which may result in inappropriate subscription states (e.g., subscriptions that are cancelled too late).

The main contributions of this paper are the following:

(i) We propose a concept of *dynamic subscription mechanisms* that improves the expressiveness and flexibility of subscriptions. To achieve this, we introduce a new dimension to a subscription that allows a subscriber to express the flow of matched events, and a condition under which a new subscription should be issued or re-issued.

(ii) We present a prototype extensible implementation of a framework called *Flexub* that supports a variation of subscription mechanisms for Publish/-Subscribe systems.

(iii) We demonstrate that a variation of three subscription mechanisms can be easily expressed in Flexub: (a) *continuous subscription*, that allows the subscriber to receive all the events that match with the subscription, (b) *subscribe once*, that allows the subscriber to receive one and only one matched event, and (c) *subscribe once and again*, that allows the subscriber to re-subscribe, if needed, after each matching.

When compared to existing subscription models, our experiment results show that the support for *dynamic subscription mechanisms* greatly reduces network traffic of events sent from publishers to the subscribers. In addition, our approach reduces the workload on the subscriber side by ensuring that subscriptions that become irrelevant are cancelled as early as possible.

2 A Motivating Scenario

Nowadays in public places such as an airport, there are many people who have mobile devices equipped with wireless communication capabilities. Such places create an opportunity for people to share information with each other without the need for fixed infrastructure. For example, the passengers in an airport can find partners to share a cab with, exchange weather information, touristic tips, etc.

Consider three passengers, Alice, Bob and Carol travelling through London Heathrow Airport. Alice is a passenger that is flying indirectly and is waiting in the airport for transit. She has a few hours between her flights and therefore, decides to find other passengers that would like to join in for a game in the lounge. She would like to share this information and *continuously* get notified whenever there is an interested passenger. Bob wants to go for vacation to an island. He arrives at the airport and has two hours before his flight. He is curious about the weather situation in the island. He would like to get notified only *one time* when there is a passenger who shares that information. Carol has just landed at the airport and wants to get a cab to her hotel. In order to minimise costs she would like to find other passengers going to the same hotel to share a cab with. She would like to get notified whenever another person is interested in sharing until the cab's capacity is reached. If the cab's capacity is not yet reached, she would like to get notified *once again* about the next person who is interested.

2.1 Scenario Analysis

The above scenario reveals interesting subscriptions that a subscriber may require in order to control the flow of information from the publisher to the subscriber. For instance, we can identify the following subscriptions:

S.1 Alice needs to issue a subscription that *continuously* matches the publications containing information about people interested in playing a game and notifies her.
S.2 Bob needs to issue a subscription that matches with *only one* publication containing weather information and notifies him.
S.3 Carol needs to issue a subscription that *once and again* matches with a publication containing information about people interested in sharing a cab and notifies her. In addition, she specifies a condition that the subscription should be reissued as long as the cab capacity is not yet reached.

These sample subscriptions highlight the need for dynamic subscriptions i.e., subscriptions that allow the subscriber to specify the flow of matched events as well as conditions under which a subscription may need to be adapted or cancelled. The current solutions suffer from limited expressiveness and flexibility and require the programmer to formulate ad hoc implementations to manage subscriptions (e.g., taking care of when to cancel a subscription or re-issue a subscription). Managing subscriptions in an ad hoc manner is error-prone and may lead to subscriptions to be cancelled too early or too late. We further discuss existing solutions in Section 5.

The next section presents our approach for expressing subscriptions with more flexibility.

3 Dynamic Subscription Mechanisms

We propose the concept of *dynamic subscription mechanisms* that improves expressiveness of subscriptions and allows the flexibility to control the flow of

matched events. The model allows the subscriber not only to express the way events should be matched but also a means to control the flow of events (e.g., a condition under which a subscription should be (re)issued or cancelled). We achieve this by introducing a new dimension of a subscription function *Si* to every subscription. In our model, subscription can be informally described as follows:

$$SUBSCRIBE(T, Si, u) \tag{1}$$

The above definition shows a subscription to an event type T, with a subscription function *Si*, and an event filtering function u. The subscription function (dimension) allows the subscriber to express extra specification about the flow of matched events, and a condition under which a new subscription can be issued or cancelled. We give a general description of the subscription dimension as follows:

$$Si = (sm, p) \tag{2}$$

Where *sm* denotes a subscription mechanism while p denotes a predicate function that specifies when to re-issue or cancel a subscription. A key observation here is that, introducing a new dimension to a subscription, makes it possible to express a subscription mechanism that specifies the semantics of the flow of matched events. So far, we have identified a variation of three subscription mechanisms: *continuous subscription*, *subscribe once*, and *subscribe once and again* that we explain below.

Continuous subscription. In this mechanism, the subscriber *continuously* receives all new matched events from the publishers. The current Publish/Subscribe systems support this kind of subscription mechanism. One drawback of this mechanism is that the publisher is always in control of the information flow, which can result in the subscriber being flooded with new information in case the publisher produces events at a higher rate than the subscriber can process.

Subscribe once. The *subscribe once* mechanism enables the subscriber to register interest in receiving a single event from the publishers and the subscription is automatically cancelled as soon as the first matched event is received. Therefore, the subscriber is in control of the information flow.

Subscribe once and again. This mechanism is a hybrid of the *subscribe once* and the *continuous subscription* mechanisms. The subscriber can receive one or multiple events depending on the need. Each subscription is specified with a predicate that is checked every time the subscriber receives an event to determine whether a subscription should be re-issued or automatically cancelled. The main advantage of this subscription mechanism is that the subscriber is in control and decides when to receive new events.

4 Flexub

In this section, we present a prototype implementation of a framework called *Flexub* that supports a variation of subscription mechanisms for Publish/Subscribe systems.

4.1 The Flexub Architecture

The Flexub architecture is composed of two layers, namely, a *communication middleware* layer, and a *language abstractions* layer, that we explain below.

Communication middleware layer. We chose Fadip [8] as our communication middleware. Fadip is lightweight Publish/Subscribe middleware that is tailored to work in MANETs. Fadip supports a hybrid model for routing by using bounded subscription propagation and message propagation at the same time. Moreover, it uses a *fading gossip* technique, which decreases the fanout of subscription and message propagation at each hop level. This technique ensures that there are less redundant message propagations in the network due to receiving the same message from different routes.

Language abstractions layer. The language abstractions layer provides high-level abstractions for expressing the dynamic subscription mechanisms supported by Flexub. For each subscription mechanism, this layer provides the corresponding high-level language abstraction. Such high-level abstractions alleviate the programmer from the burden of manual management of subscription cancellation and event filtering. Managing subscriptions in such an ad hoc manner is error-prone and may lead to subscriptions to be cancelled too early or too late. The details of using our language abstractions to implement the airport scenario can be found in [1].

4.2 Evaluation: Simulated Experiment

We have performed a preliminary experiment in order to evaluate the benefits of adding support for dynamic subscription mechanisms to a Publish/Subscribe system. The goal of this experiment was to investigate two properties: (i) the effect of dynamic subscription mechanisms on the network traffic of matched events, and (ii) the workload on the subscriber side i.e., the amount of matched events that are received by a subscriber versus the amount of events that are required by the subscriber.

4.2.1 Experiment Setup

We implemented Flexub on top of Fadip [8], to investigate the effects of using a dynamic subscription mechanism to control the flow of events. We performed the simulations using OMNeT++ [11], a component-based, open-architecture discrete event network simulator that is widely used in academia for the simulation of computer networks. We also used an OMNeT++ modelling framework called MiXiM to simulate wireless communication in the vehicular network.

We used 50 to 150 nodes in the simulations. We randomly chose 20% of these nodes as publishers and 5% of them as subscribers. The simulation runs for 50 seconds and nodes move randomly around with a speed of 2 meters per second in a playground with a size of 1500 meters by 1000 meters. With this setting, we tried to resemble the movement of people walking around with their

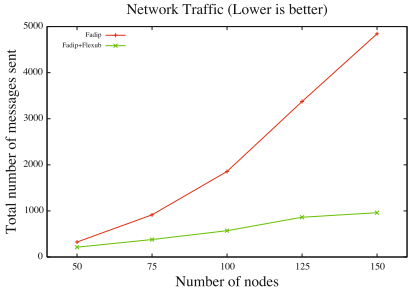

Fig. 1. Network traffic usage by the number of nodes, with and without using Flexub

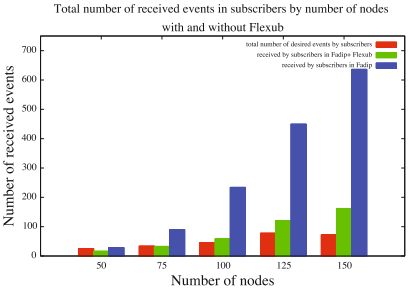

Fig. 2. The number of desired events and the number of received events by subscribers in Fadip and in Fadip+Flexub

computational devices in an airport terminal. Each subscribing node sends its subscription in the beginning of the simulation and each publishing node sends an event every 5 seconds.

First, we performed the simulation without using Flexub. In this setting, the subscriptions are propagated without any flow control, and they keep continuing to match and forward events. In the second setting, we use Flexub to add flow control to the subscriptions. We assign a random number between 1 to 10, with a normal distribution, that indicates the number of matched events, after which the subscription is automatically cancelled.

4.2.2 Network Traffic

Figure 1 depicts the effect of using Flexub on network traffic usage, with a network size of 50 nodes up to 150 nodes. We measure network traffic usage by counting the total number of messages that all nodes sent during the simulation time. This includes both subscription and publication messages. The results suggest that:

- Flexub reduces the network traffic usage considerably. The reason is that in Flexub, subscriptions have a flow control mechanism, and after the desired amount of matched events for a subscription has been reached, they prevents other unnecessary events to be further propagated in the network.
- Flexub helps Fadip to scale better. Without Flexub, the network traffic usage increases exponentially by increasing the number of nodes. However, Flexub linearises this effect on network traffic. This means that the support for dynamic subscriptions makes a Publish/Subscribe system to be more scalable and to work more efficiently with the increase in the number of nodes and messages on the network.

4.2.3 Workload on the Subscribers

Figure 2 illustrates the effect of using Flexub on the number of events that subscribers receive, comparing to the total amount of desired matched events.

The number of desired events is the sum of the required matched events for every subscriber in the network. Moreover, the number of received events is the sum of all events that a subscriber receives that can be more than what the subscribers initially asked for. As the results suggest, Flexub helps to the subscribers to receive fewer extra matching events. This reduces the workload on the subscribers. The reason is that Flexub automatically cancels the propagation of extra matched events in the early stages of their propagation. The benefit of this effect is reducing the processing effort in the subscribers.

5 Related Work

In [4] the authors propose Cayuga that aims to extend the expressive power of Publish/Subscribe systems by allowing subscriptions that span multiple events and supporting parameterisation and aggregation of events. However, in Cayuga it is not possible to express dynamic subscriptions and it does not provide support for specifying conditions under which a subscription can be automatically cancelled.

Jayaram *et. al.* [7] propose parametric subscriptions in order to provide support for dynamic subscription adaptations. They enrich content-based Publish/-Subscribe system with the ability for the subscriber to dynamically update its subscription without loosing any events that occur between the adaptations. Their work can be complementary to our approach in order to efficiently change from one subscription mechanism to another.

Drosou *et. al.* [5] propose a preference-aware Publish/Subscribe system with a ranking mechanism based on the user preferences. Their system only delivers the top-ranked events to the subscribers, and filters out the rest. Therefore, they aim to increase the relevance of the events received by the users. In comparison with our system, they focus on the relevance of the matched events and not in controlling the flow of the events.

6 Conclusions and Future Work

We proposed the *dynamic subscription mechanisms* concept that improves the expressiveness and flexibility of subscriptions. Our preliminary experiments show that the adding support for dynamic subscriptions greatly reduces the network traffic of matched events received by subscribers. In addition, our approach reduces the workload on the subscriber side by ensuring that subscriptions that become irrelevant are cancelled as early as possible.

For future work, we plan to investigate subscription mechanisms that reduce the workload at the publisher side as well as other intermediary nodes in a Publish/Subscribe system.

Acknowledgements. We would like to thank Pieter Mensalt for the initial experiments of subscription mechanisms in iScheme [2]. This work was partially

funded by the SAFE-IS project of the Research Foundation - Flanders (FWO), the STADiUM project, and the VariBru project of the ICT Impulse Programme of the Institute for Research and Innovation.

References

1. Bainomugisha, E., Paridel, K., Vallejos, J., Berbers, Y., Meuter, W.D.: Flexub: Dynamic subscriptions for publish/subscribe systems in manets. Technical Report VUB-SOFT-TR, Vrije Universiteit Brussel, Belgium (April 2012), `http://soft.vub.ac.be/~ebainomu/publications/flexub-tr.eps`
2. Bainomugisha, E., Vallejos, J., Boix, E.G., Costanza, P., D'Hondt, T., De Meuter, W.: Bringing Scheme programming to the iPhone–Experience. Software: Practice and Experience (2011)
3. Bittner, S., Hinze, A.: Pruning subscriptions in distributed publish/subscribe systems. In: Proceedings of the 29th Australasian Computer Science Conference, ACSC 2006, vol. 48, pp. 197–206. Australian Computer Society, Inc., Darlinghurst (2006)
4. Demers, A., Gehrke, J., Hong, M., Riedewald, M., White, W.: Towards Expressive Publish/Subscribe Systems. In: Ioannidis, Y., Scholl, M.H., Schmidt, J.W., Matthes, F., Hatzopoulos, M., Böhm, K., Kemper, A., Grust, T., Böhm, C. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 627–644. Springer, Heidelberg (2006)
5. Drosou, M., Stefanidis, K., Pitoura, E.: Preference-aware publish/subscribe delivery with diversity. In: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, p. 6. ACM (2009)
6. Fabret, F., Jacobsen, H.A., Llirbat, F., Pereira, J., Ross, K.A., Shasha, D.: Filtering algorithms and implementation for very fast publish/subscribe systems. In: Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, SIGMOD 2001, pp. 115–126. ACM, New York (2001)
7. Jayaram, K.R., Jayalath, C., Eugster, P.: Parametric Subscriptions for Content-Based Publish/Subscribe Networks. In: Gupta, I., Mascolo, C. (eds.) Middleware 2010. LNCS, vol. 6452, pp. 128–147. Springer, Heidelberg (2010)
8. 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)
9. Petrovic, M., Muthusamy, V., Jacobsen, H.: Content-based routing in mobile ad hoc networks. In: The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 45–55. IEEE (2005)
10. Taherian, S., Bacon, J.: State-filters for enhanced filtering in sensor-based publish/subscribe systems. In: Proceedings of the International Workshop on Data Intensive Sensor Networks (DISN 2007), Mannheim, Germany. IEEE Press (May 2007)
11. Varga, A., et al.: The OMNeT++ discrete event simulation system. In: Proceedings of the European Simulation Multiconference (ESM 2001), pp. 319–324 (2001)