Self-Organizing Publish/Subscribe

Extended Abstract

Michael A. Jaeger
Communication and Operating Systems Group
Institute for Telecommunication Systems
Technical University of Berlin, Germany
michael.jaeger@acm.org

ABSTRACT

The increasing availability of broadband Internet access coming along with cheap flatrate tariffs changed the way the Internet is used over the last years. As a result, we observe more users that become producers and collaborate instead of solely consuming information as they did before. Besides that, the Internet has become a tool for daily use and information dissemination. As push-based information services are heavily needed for collaboration and information dissemination, it seems like the breakthrough of distributed push publish/subscribe on the open Internet is just imminent today, although research has been around already for many years now. One reason why most publish/subscribe applications are still based on a rather basic centralized mechanism and not on a distributed scalable notification service, as it is commonly proposed in state-of-the-art publish/subscribe systems in research, is the unsolved management issue.

This extended research abstracts is intended to give an overview of my Ph.D. project on self-organizing publish/subscribe, where I want to tackle management issues for better applicability. After giving an introduction to the pub/sub communication paradigm and motivating the necessity to make such systems self-organizing, I will point out the contributions of my work. They are centered around making publish/subscribe systems self-stabilizing and adaptive by means of self-organizing mechanisms. While my work on self-stabilization is nearly finished, introducing adaptivity by self-organization is still in an early stage.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed applications*; C.4 [**Computer Systems Organization**]: Performance of Systems—*Fault tolerance*

2nd International Doctoral Symposium on Middleware '05, November 28- December 2, 2005 Grenoble , France

General Terms

Algorithms, Management, Reliability

Keywords

Publish/Subscribe, Self-Stabilization, Adaptivity, Self-Organization

1. INTRODUCTION

Publish/subscribe systems can be one of the building blocks for next generation distributed applications including social software and the increasing pervasion of our daily life with networked computers. Therefore, the brittle structure of currently used systems has to be made more flexible for better applicability and reduced management costs. Today, we see a lot of applications that are inherently based on collaboration and fast information dissemination. Scalability and the demand for efficient push mechanisms characterize these applications. Although there are already publish/subscribe applications like RSS successfully deployed on the Internet, they use rather basic mechanisms and suffer from scalability problems.

The *publish/subscribe* (pub/sub) communication paradigm best suits information-driven applications (*e.g.*, news and update services). Those applications can be implemented in many ways, either *centralized* or *distributed* and either *push* or *pull*. In case of distributed push (which is the technique considered in this paper), it is common to use an overlay-network of brokers providing a *notification service*. This approach is more scalable for obvious reasons. Clients connect to their *local broker* and are able to *publish* or *subscribe to* notifications. The notification service then takes care that every notification published is forwarded to those brokers that have local clients connected to them that subscribed to this notification. An example topology of a distributed notification service is depicted in Figure 1.

There has been a lot of research in the area of routing algorithms and on scalability issues, while the topology of the notification service in environments like the Internet is mostly assumed to be either static or manually managed. In practice, usage patterns in the system are highly dynamic for many applications. Consider, for example, a global network delivering stock quotes: it is obvious that the number of notifications concerning stocks traded at the New York Stock Exchange suddenly drops when it closes. Another scenario could be an e-home environment, where all networked devices connect through a publish/subscribe system. Here,

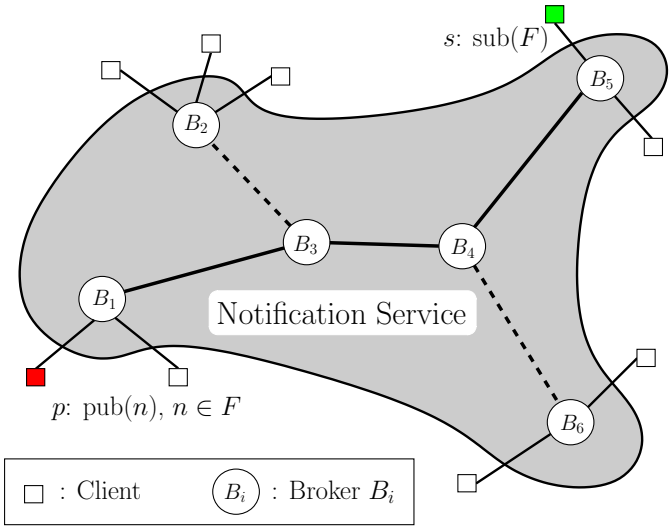

Figure 1: Example for a publish/subscribe topology where client p publishes a notification n that is matched by filter F, client s subscribed to, while the notification service takes care of forwarding the notification properly.

it is highly probable that nodes join and leave the network now and then. In addition to that, it is reasonable to assume that no skilled administrator is available to take care of the system configuration. Taking situations like these into account, it is sensible to search for mechanisms such that the broker topology is able to adapt itself to current usage patterns and failures in the network and optimize its quality of service thereby.

Besides efficiency and quality of service, fault-tolerance is a severe issue as pointed out in the e-home scenario. There has already been some research focused on fault-tolerance issues in publish/subscribe systems. Some approaches employ a fault-tolerant peer-to-peer routing substrate on which the pub/sub middleware implementation is built upon [19, 23], others focus on unstable mobile scenarios where links vanish spontaneously [18]. In [3], Baldoni et al. use a heartbeat mechanism to detect failed brokers. In these approaches, only broker or link failures are considered. Corrupted data structures are not taken into account, for example. An interesting approach based on gossiping has been proposed in [5] that tries to overcome the problem of lost messages. I am planning to add deterministic and provable self-stabilization features to publish/subscribe systems.

The combined use of self-organizing and self-stabilizing mechanisms in the notification service is another concern, as it has to be guaranteed that they coexist peacefully. Here, I hope to find ways to efficiently combine both mechanisms to enhance the benefit for the performance of the system.

2. CONTRIBUTIONS

The contributions of my thesis will be on two main objectives:

1. further exploring and analyzing *self-stabilizing pub/sub* systems and
2. making them *self-organizing* through mechanisms that only rely on local information and reconfiguration, while trying to improve the system performance in whole.

Self-stabilization can be realized separately in two different ways on two different layers, namely the overlay network and the pub/sub routing layer, while self-organization of the broker topology based on the higher layer dynamics concerns both layers at once, as reorganizing the broker topology implies a reconfiguration of the routing layer. It is a standard technique to layer self-stabilizing algorithms to compose a certain self-stabilizing behavior, which is unproblematic as long as the different layers have no cyclic dependencies [8].

As self-organization has to be layered on top of the self-stabilizing mechanism it is important to find out, if and how both mechanisms can coexist without disturbing each other. The different layers are depicted in Figure 2.

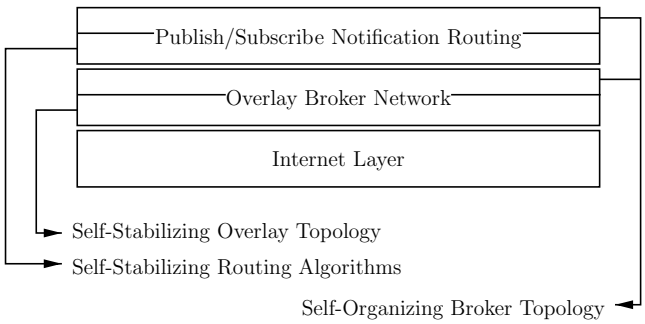

Figure 2: Contributions on the different layers of a publish/subscribe system.

3. SELF-STABILIZATION

The principle of *self-stabilization* has been introduced by Dijkstra in 1974 [7]. The idea is, that a system pushes itself—no matter what its current state is like—to a legitimate (*i.e.*, correct) state in a bounded number of steps (*convergence*). Once a correct state has been reached, a self-stabilizing system stays there, provided that no failures occur (*closure*) [1]. Both properties are only *guaranteed* if failures are temporary (*transient*). This principle has been successfully applied to several problems before (*e.g.*, networking protocols) and is very attractive for lowering management and implementation costs as almost no assumptions regarding failures are necessary. On the downside, no guarantees concerning the behavior of the system can be made while it is not in a legitimate state. Realizing a self-stabilizing pub/sub system is an important step for applying this communication paradigm in areas where administration is a concern but service unavailability is passable, for example, in networked e-home environments where it is likely that no person is available that is capable of taking care of administration tasks.

In pub/sub systems, my focus is on on making notification routing self-stabilizing as well as layering it on top of a self-stabilizing overlay network. The first work on self-stabilizing routing has been published by Mühl in his dissertation which I used as a starting point for my research [15]. The only other work in the area of self-stabilizing publish/subscribe I am aware of is the one of Shen and Tirthapura who try to make pub/sub routing self-stabilizing using Bloom filters to compare the contents of routing tables of different brokers [21]. The problems that come up with this approach are discussed in [16].

My work on self-stabilization in publish/subscribe systems is nearly finished. The basic algorithms are proposed in [16] and rely on a leasing mechanism: subscribers in the system regularly refresh their subscriptions every refresh period ρ. The value of the refresh period determines the leasing period π that parameterizes a second-chance algorithm running on every broker which removes routing table entries that have not been renewed in time. Both values depend on the desired stabilization time, which again depends on the size of the network and the maximum link delay. Using a leasing mechanism introduces additional control traffic to the network which might be inefficient in certain settings. To be able to reason about the efficiency, we compared the self-stabilizing algorithms proposed with flooding which is self-stabilizing by nature and compared the messages saved by using advanced routing algorithms with the control traffic added. Using simulation we were able to find the break-even point to prefer flooding if an average number of subscriptions in the system has been reached and subscriptions are equally distributed on the brokers and the filter classes. Simulation is time consuming and so we took an analytical approach to evaluate the algorithms using Markov Chains. As a result, we found a formalism for calculating the message complexity for certain settings [14].

Up to now, we have not considered the incorporation of manual reconfiguration in our scenarios. With respect to the overlay network we assumed that it is either static or itself self-stabilizing. In practice, simply layering self-stabilizing routing on top of a self-stabilizing overlay network is somehow problematic if administrators manually change the configuration (*e.g.*, for maintenance reasons). An intervention like this is viewed as a failure on the self-stabilizing overlay network layer and may lead to a reconfiguration of the overlay network, which again may lead to a reconfiguration on the publish/subscribe routing layer. Although this is in perfect accordance with the condition that during reconfiguration no guarantees are given for a self-stabilizing system, this may lead to degraded system performance even though no "real" fault happened.

The problems that arise with layering self-stabilizing routing on top of a self-stabilizing broker overlay has not been investigated yet. Reconfiguration in publish/subscribe systems has been worked on mostly with focus on mobile environments [6, 18] with the underlying assumption, that links vanish in an unpredictable manner. Here, the authors did not care explicitly about message ordering which is an important issue in many applications. Recently, Tarkoma et al. published work on subscription and advertisement completeness in mobile environments [22]. Another approach is layering publish/subscribe routing on top of a structured peer-to-peer routing substrate, inheriting the fault-tolerance mechanism of the lower layers this way [4, 19, 23]. In contrast to my approach, this approach is fault-tolerant and fault-masking in many situations but not self-stabilizing in general. Additionally, reconfiguration is only considered on the physical network level and not on the overlay network. Recently, Frey and Murphy proposed several algorithms for maintaining a publish/subscribe overlay tree in large-scale dynamic networks [10]. Again, this paper focuses on keeping the overlay network connected and maintaining an acyclic graph while leaving ordering aside.

I'm currently working on combining manual reconfiguration with self-stabilizing overlay networks in publish/subscribe. Here, I'm thinking about applying the concepts of *super-stabilization* [9] and *fault-containment* [13] to improve applicability.

4. SELF-ORGANIZING MECHANISMS

Endowing publish/subscribe systems with *self-organizing* capabilities to enhance adaptivity in dynamic environments can be done with respect to the structure of the overlay broker network that forms the notification service. Related work in this area either relied on the self-organizing mechanisms provided by a peer-to-peer routing substrate [4, 19, 23] or tried to reorganize the overlay network according to broker "interests" [2, 3, 24]. The first approach, although sensible at the first sight, has the disadvantage, that the structure of the overlay network is totally agnostic of the routing configuration in the upper pub/sub layer. Thus, the risk remains, that the evolution of the overlay network leads to sub-optimal performance.

The latter approach is far more sophisticated as it tries to perform kind of a cross-layer optimization between the overlay network and the pub/sub routing layer. Virgillito et al. approach the problem of determining similar interests between brokers by calculating the intersections of the filters they store in their routing tables: the greater the intersection between two brokers is, the bigger is the common interest (*associativity*), and thus the fewer TCP-hops should be on the path between those two. There are some problems with this approach. The first is that calculating the common interest is sometimes very hard and expensive, depending on the filter model used. If it is possible at all and can be implemented in an efficient manner, the implementation is only suitable for this particular filter model, the algorithm has been designed for, and still it is not quite clear if the associativity can really be used to forecast the number of identical notifications consumed as it might be that never any notification is published in the intersected area of interest. Another issue is, that the cost metric used is rather simple as it only takes overlay TCP-hops (and some network level metrics in an extension) into account. The explicit target is to reduce the number of pure forwarder brokers (*i.e.*, brokers that only forward a notification they receive to neighbor brokers and that do not have any local client that has subscribed for it). This way, it is not taken into account that forwarders can be successfully used to decrease network traffic and thus increase system performance. The authors argue that a lower number of TCP-hops leads to fewer matching operations which is certainly true but not

the pressing problem if processor load is not the bottleneck but message complexity is. Additionally, the reconfiguration that takes place when links are relocated does not use standard pub/sub system operations which makes it more intrusive to implement. Finally, correctness properties during reconfiguration are not considered. As this work represents the first step towards self-organizing pub/sub systems, there is still a lot of room for improvements.

The idea of adapting the broker topology to usage patterns stems from the insight that connecting brokers that consume a lot of identical notifications can lead to lower communication and routing costs in the broker network. Previous work presented before concentrated only on the similarities of subscriptions different brokers manage, leaving aside the fact that the actual traffic that two brokers consume is not necessarily correlated to the filters they manage. Our approach considers the actual message flow to decide if and how a reconfiguration of the overlay network can help to improve system performance. Therefore, brokers compare the notifications they consumed with each other in an n-hop neighborhood to find out if there is another broker that consumed many identical notifications. According to a cost metric that takes the processor load into account, the brokers try to reach a consensus if and how a reconfiguration can be performed, as a reconfiguration might lead to increased performance for one broker but a big decrease for many others. Finally, the reconfiguration will be performed while notifications are queued in the critical section to maintain message completeness and ordering properties.

Routing algorithms for publish/subscribe systems are well understood nowadays and it is clear that costs for maintaining routing tables and matching notifications is a trade-off to keeping the message complexity and message delay low. However, it is obvious that depending on the state of the system it could be sensible to prefer one routing algorithm in favor of another and to switch them dynamically or tune parameters accordingly (*adaptive filtering*). In the easiest case, for example, if there are subscribers nearly everywhere for most of the notifications published, it might be advisable to switch the routing algorithm used from an advanced algorithm to flooding, saving computation and memory cost as well as the traffic needed for the management of routing tables in favor of an increased message complexity. Deciding when to switch or how to tune parameters (*e.g.*, broadening filters) and which routing algorithm to choose, as well as actually performing the switch, will be another contribution of the thesis. One hard problem is here to draw decisions based solely on local information.

5. FUTURE WORK

The work on basic self-stabilization in publish/subscribe systems is nearly finished. Currently, I'm working on manual reconfiguration in self-stabilizing publish/subscribe systems as well as generalizing the stochastic analysis published in [14]. Another issue in this field regards fault-masking which would be interesting to combine with self-stabilization.

Regarding self-organizing publish/subscribe systems I currently work on the metric used to decide if a reconfiguration should take place and the actual reconfiguration while maintaining basic correctness properties. Here, the cost of the reconfiguration itself is still an open issue as well as problems that might come up with oscillation.

Because it is hard, if not impossible, to formally analyze the efficiency of the resulting algorithms, I will need to perform simulations. Therefore, it is now time to implement a simulation framework or find an existing one that suits my needs. The idea on adaptive filtering is rather new. Thus, my next steps in this field will be to further explore the possibilities of this approach.

As mentioned in the introduction, there are already publish/subscribe applications on the Internet that could profit from using a scalable publish/subscribe infrastructure. I am planning to combine state-of-the-art publish/subscribe techniques with classic RSS applications to enhance performance, functionality, and scalability. First work in this area has been published in [20, 17]. In the security application domain, we already performed successful experiments, applying pub/sub as the basic communication paradigm in attack prevention systems [11, 12].

Acknowledgements

I thank my advisor Prof. Hans-Ulrich Heiß and my colleagues Gero Mühl, Klaus Herrmann, and Matthias Werner for their scientific guidance and the many discussions we had. Last not least, I thank Deutsche Telekom Stiftung for the financial support.

6. REFERENCES

[1] A. Arora and M. G. Gouda. Closure and convergence: A foundation of fault-tolerant computing. *Software Engineering*, 19(11):1015–1027, 1993. Definition of fault-tolerance.

[2] R. Baldoni, R. Beraldi, L. Querzioni, and A. Virgillito. Subscription-driven self-organization in content-based publish/subscribe. Technical report, Midlab Middleware Laboratory (University of Rome), Mar. 2004. `http://www.dis.uniroma1.it/~midlab/docs/BBQV04ICAC.pdf`.

[3] R. Baldoni, R. Beraldi, L. Querzoni, and A. Virgillito. A self-organizing crash-resilient topology management system for content-based publish/subscribe. In *3rd International Workshop on Distributed Event-Based Systems (DEBS'04)*, Edinburgh, Scotland, UK, May 2004.

[4] R. Baldoni, C. Marchetti, A. Virgillito, and R. Vitenberg. Content-based publish-subscribe over structured overlay networks. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, 2005 (ICDCS 2005)*, pages 437–446, 2005.

[5] P. Costa, M. Migliavacca, G. P. Picco, and G. Cugola. Introducing reliability in content-based publish-subscribe through epidemic algorithms. In *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03)*, June 2003.

[6] G. Cugola, D. Frey, A. L. Murphy, and G. P. Picco. Minimizing the reconfiguration overhead in content-based publish-subscribe. In *Proceedings of the 2004 ACM symposium on Applied computing (SAC'04)*, pages 1134–1140, New York, NY, USA, 2004. ACM Press.

[7] E. W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974.

[8] S. Dolev. *Self-Stabilization*. MIT Press, 2000.

[9] S. Dolev and T. Herman. Superstabilizing protocols for dynamic distributed systems. *Chicago Journal of Theoretical Computer Science*, 4, Dec. 1997. Special Issue on Self-Stabilization.

[10] D. Frey and A. L. Murphy. Maintaining a publish-subscribe overlay tree in large-scale dynamic networks. 2005. Under review: `http://www.inf.unisi.ch/murphy/Papers/middleware.pdf`.

[11] J. García, J. Borrell, M. A. Jaeger, and G. Mühl. An alert communication infrastructure for a decentralized attack prevention framework. In *Proceedings of the IEEE International Carnahan Conference on Security Technology (ICCST)*. IEEE, Sept. 2005. Accepted for publication.

[12] J. García, M. A. Jaeger, G. Mühl, and J. Borrell. Decoupling components of an attack prevention system using publish/subscribe. In R. Glitho, A. Karmouch, and S. Pierre, editors, *Proceedings of the 2005 IFIP conference on Intelligence in Communication Systems*, volume 190, Montreal, Canada, Oct. 2005. IFIP, Springer. Accepted for publication.

[13] S. Ghosh, A. Gupta, T. Herman, and S. Pemmaraju. Fault-containing self-stabilizing algorithms. In *Proceedings of the Fifteenth Annual ACM Symposium of Distributed Computing (PODC96)*, pages 45–54. ACM, ACM Press, 1996.

[14] M. A. Jaeger and G. Mühl. Stochastic analysis and comparison of self-stabilizing routing algorithms for publish/subscribe systems. In G. F. Riley, R. Fujimoto, and H. Karatza, editors, *The 13th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2005)*, pages 471–479, Atlanta, Georgia, USA, Sept. 2005. IEEE Press.

[15] G. Mühl. *Large-Scale Content-Based Publish/Subscribe Systems*. PhD thesis, Darmstadt University of Technology, Sept. 2002.

[16] G. Mühl, M. A. Jaeger, K. Herrmann, T. Weis, L. Fiege, and A. Ulbrich. Self-stabilizing publish/subscribe systems: Algorithms and evaluation. In J. C. Cunha and P. D. Medeiros, editors, *Proceedings of the 11th European Conference on Parallel Processing (Euro-Par 2005)*, volume 3648 of *Lecture Notes in Computer Science (LNCS)*, pages 664–674, Lisboa, Portugal, Aug. 2005. Springer-Verlag.

[17] M. Petrovic, H. Liu, and H.-A. Jacobsen. CMS-toPSS: Efficient dissemination of RSS documents. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 1279–1282, Trondheim, Norway, 2005.

[18] G. P. Picco, G. Cugola, and A. L. Murphy. Efficient content-based event dispatching in the presence of topological reconfiguration. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, pages 234–243. IEEE Computer Society, 2003.

[19] P. R. Pietzuch and J. Bacon. Peer-to-peer overlay broker networks in an event-based middleware. In *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03)*, pages 1–8, June 2003.

[20] D. Sandler, A. Mislove, A. Post, and P. Druschel. FeedTree: Sharing web micronews with peer-to-peer event notification. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, Ithaca, NY, Feb. 2005.

[21] Z. Shen and S. Tirthapura. Self-stabilizing routing in publish-subscribe systems. In *3rd International Workshop on Distributed Event-Based Systems (DEBS'04)*, Edinburgh, Scotland, UK, May 2004.

[22] S. Tarkoma and J. Kangasharju. Mobility and completeness in publish/subscribe topologies. In *International Conference on Networks and Communication Systems (IASTED)*. ACTA Press, Apr. 2005.

[23] W. W. Terpstra, S. Behnel, L. Fiege, A. Zeidler, and A. P. Buchmann. A peer-to-peer approach to content-based publish/subscribe. In *Proceedings of the 2nd international workshop on Distributed event-based systems (DEBS'03)*, pages 1–8, New York, NY, USA, 2003. ACM Press.

[24] A. Virgillito. *Publish/Subscribe Communication Systems: From Models to Applications*. PhD thesis, Universita La Sapienza, 2003.