Publish/Subscribe-based Cluster and Routing Algorithm for Actuator and Sensor Networks

Jan Hendrik Schönherr[*]
schnhrr@cs.tu-berlin.de

Helge Parzyjegla
parzyjegla@acm.org

Gero Mühl
g_muehl@acm.org

Communication and Operating Systems Group
Berlin Institute of Technology
Einsteinufer 17 EN6, 10587 Berlin, Germany

ABSTRACT
In this paper we present a self-organizing and self-stabilizing cluster and routing algorithm for actuator and sensor networks that is solely based on publish/subscribe primitives. The algorithm design captivates with its simplicity and is yet robust against node and link failures. The provided publish/subscribe middleware which incorporates the algorithm allows an elegant application design for actuator and sensor networks.
Keywords: Publish/Subscribe, Actuator and Sensor Networks, Cluster, Self-Stabilization, Self-Organization

1. INTRODUCTION
Since the miniaturization of electronic devices advances rapidly, in the near future, a person will habitually carry a whole lot of intelligent gadgets that unobtrusively provide assistance in daily work and life. Moreover, the environment itself will be equipped with a multitude of electronic sensors and actuators to interact with. In order to support meaningful applications all these devices have to communicate (wirelessly) by forming complex actuator and sensor networks (AS-Nets). AS-Nets need to be self-organizing as normal users cannot be expected to be able or even willing to administer a dynamic network consisting of possibly hundreds of devices [6]. Thus, the network should ideally handle all configurations, adaptations, and failures itself. Considering the complexity of the network, its dynamic nature, and existing fault tolerance requirements, it is also not sensible to rely on a single controlling instance having global knowledge to carry out all administrative tasks. Instead, devices have to cooperatively contribute to network management.

An appealing way to organize cooperation is to employ an event-driven style of interaction by exploiting publish/subscribe: *producers* publish notifications, while *consumers* selectively subscribe to notifications, for example, using topic-based or content-based filters [5]. Publish/subscribe ideally fits the targeted setting because, usually, neither producers nor consumers do address their counterparts explicitly. This leads to a loosely and dynamic coupling of components allowing for more flexibility and fault resistance than if an explicit addressing of individual nodes was used.

In this paper we present a self-organizing cluster and routing algorithm for AS-Nets that builds upon publish/subscribe primitives. Publish/subscribe is used for cluster formation as well as for intra-cluster and inter-cluster communication providing a lightweight middleware layer for event-driven applications. Addressing clustering and event routing with the same means in an integrated approach reduces the complexity considerably compared to combining other approaches that focus either on clustering [7] or on routing [4].

2. CLUSTERING PUB/SUB ALGORITHM
Our algorithm has very moderate prerequisites. We only require a simple radio network with basic broadcast capabilities. In the following we first describe the cluster formation and, then, the cluster maintenance.

2.1 Cluster Formation
At first, basic k-hop clusters are formed using a heartbeat that each node broadcasts periodically. Therefore, a heartbeat contains information about the node's cluster and its distance to the cluster-head measured in hops. A new node listens to heartbeat signals and joins any suitable cluster by simply announcing its membership in its own heartbeat. If no suitable heartbeat is received within a certain amount of time the node starts to announce itself as the head of a new cluster. In each cluster, the nodes set up a publish/subscribe broker network based on the shortest path tree to the cluster-head which is established as a byproduct of the heartbeat signal. This is shown as level 0 in Fig. 1.

After setting up intra-cluster communication, the broker network is utilized to realize inter-cluster communication: The cluster-head subscribes for *inward notifications*, while gateway nodes (nodes that can hear heartbeats from different clusters) place appropriate subscriptions for *outward notifications* in their own cluster. Gateway nodes also have the ability to publish notifications in neighboring clusters. This enables a message flow between cluster-heads of adjacent clusters: A cluster-head publishes a message wrapped in an outward notification, the gateway node re-wraps this

[*]Funded by Deutsche Forschungsgemeinschaft.

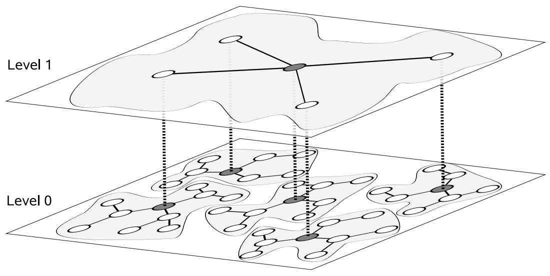

Figure 1: Schematic view of the created broker topology. Level 0 shows existing physical nodes and links. Level 1 (and up) is logical only.

message into an inward notification and publishes it in the destination cluster, where it will be consumed by the cluster-head. In order to ensure that only one of the suitable gateway nodes forwards a message, the primitive publishSingle is used that ensures that a messages is only delivered to a single consumer.

This enables us to realize communication primitives on a cluster to cluster basis (namely a broadcast and a directed send). This in turn allows us to apply our algorithm recursively, as this upper level can be handled in exactly the same way as the level below: sending heartbeats, forming clusters, setting up inter-cluster communication and so on. A possible result of this upper level is shown in level 1 in Fig. 1. This process is repeated until the whole network is covered by a single cluster on the highest level. Global publish/subscribe functionality for applications is easily achieved by allowing normal (i. e., not cluster related) subscriptions to cross level boundaries (depicted as dotted lines in Fig. 1).

2.2 Cluster Maintenance

All information that is kept in the network is periodically refreshed, making it robust against sporadic message loss, duplication, mutation, and also transient memory corruption. Therefore, our approach is self-stabilizing; an important property in unsupervised networks [1].

Node failures or changes in the network topology are detected by the absence of heartbeats. They are handled on the lowest level possible: as long as a failure or change can be handled locally in a cluster, it is not propagated to the next level. Link failures are automatically reflected by vanishing subscriptions in the publish/subscribe substrate. As these subscriptions are inherent redundant, there is no need to reconstruct routes between clusters. If no route is left to a cluster, this is handled as a link failure on the next higher level. If either the cluster gets partitioned or it loses too much internal state (currently this would be a cluster-head failure), the remaining nodes reassign themselves to different clusters if possible, otherwise new clusters are generated. Because of this we do not have a ripple effect that might cause a reorganization of the complete network that occurs for many self-stabilizing algorithms.

The infrastructure repairing algorithms are intentionally very simple. We do not utilize a special handling for each and every situation. Naturally this comes along with increased self-stabilization times. To overcome this, we plan to introduce adaptive heartbeats: if a failure or a change is detected, the normal heartbeat period is shortened near the origin, so that the normal (and simple) algorithms perform their work faster. We consider this and the avoided ripple effect to be forms of fault containment [3] and superstabilization [2].

3. CONCLUSIONS AND FUTURE WORK

We presented a new approach to routing and clustering in wireless AS-Nets, which impresses with its simplicity. Application developers are provided with a publish/subscribe middleware, which is deeply integrated into the communication mechanism, and, thus, avoids overhead that would normally occur when realizing a publish/subscribe layer on top of a usual routing protocol. Using publish/subscribe as a routing mechanism in wireless networks reveals several synergies, such as automatic route recovery that must be explicitly developed in traditional approaches.

While our approach introduces a certain overhead because of the hierarchical clustering, we believe that this is outweighed by the gained amount of fault-tolerance. In the future we will address this overhead by reducing the special tasks of cluster-heads and explore the versatility of our approach by exchanging the used clustering and publish/subscribe substrates. Furthermore, we will perform an in depth evaluation of the approach to confirm its usefulness in the use cases we target at.

4. REFERENCES

[1] S. Dolev. *Self-Stabilization.* MIT Press, Cambridge, MA, USA, 2000.

[2] S. Dolev and T. Herman. Superstabilizing protocols for dynamic distributed systems. *Chicago Journal of Theoretical Computer Science*, 1997(4):1–40, Dec. 1997. Special Issue on Self-Stabilization.

[3] S. Ghosh, A. Gupta, T. Herman, and S. V. Pemmaraju. Fault-containing self-stabilizing algorithms. In *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC '96)*, pages 45–54, New York, NY, USA, May 1996. ACM Press.

[4] X. Hong, K. Xu, and M. Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE Network*, 16(4):11–21, July/Aug. 2002.

[5] G. Mühl, L. Fiege, and P. R. Pietzuch. *Distributed Event-Based Systems.* Springer, Aug. 2006.

[6] H. Parzyjegla, M. A. Jaeger, G. Mühl, and T. Weis. A model-driven approach to the development of autonomous control applications. In *Proceedings of the 1st Workshop on Model-driven Software Adaptation (M-ADAPT '07) at ECOOP 2007*, pages 25–27, Berlin, Germany, July 2007.

[7] J. Y. Yu and P. H. Chong. A survey of clustering schemes for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, 7(1):32–48, 2005.