Challenges in Content Based, Semantically Decoupled Communication on Neighbor-Relations

Henry Ristau

University of Rostock, Faculty of Computer Science and Electrical Engineering, Institute of Computer Science, Information and Communication Services Group, 18051 Rostock, Germany
henry.ristau@uni-rostock.de

Abstract. Announcement/Subscription/Publication (ASP) is an approach for content-based communication, decoupled in time, space, threads and semantics. The idea behind ASP is to rely only on neighbor-relations and does not involve any communication infrastructure above the link-layer. This makes the ASP approach perfectly suitable for application communication in smart and ubiquitous environments. In this paper we analyze the process of implementing scenarios using the ASP approach. After specifying the necessary requirements for application interfaces and the behaviour of the middleware we identify circumstances where problems could arise and present our solutions to these problems.

1 Introduction

Future smart and ubiquitous environments emerge from the ad-hoc cooperation of different devices surrounding the user in her everyday life. The goal of such cooperation is to support the users in what they are doing e.g. by enriching her environment with information, controlling parts of her environment to suit her needs or providing her with services to support her daily routine.

A main requirement for this kind of cooperation is the ability of applications on these devices to exchange information. The aforementioned ubiquitous environments however often have a very heterogeneous nature in terms of devices and communication techniques. Furthermore their ad-hoc generated topology can not be expected to provide compatible communication protocols or even unique addresses. This results in the strong need for a middleware to support decoupled communication in space, time and threads [1][2][3].

Publish/Subscribe has emerged as a paradigm to support the distribution of information in a decoupled way. However especially in ad-hoc generated smart environment very heterogeneous applications are to be expected. Applications provide and seek for information in different formats and levels of aggregation. This often requires information to be aggregated and processed while it is delivered from source to sink and results in the need for decoupling in semantics as well.

D. Tavangarian et al. (Eds.): IMC 2009, CCIS 53, pp. 189–200, 2009.

Announcement/Subscription/Publication (ASP) [4] is an approach to provide content based communication decoupled in space, time, threads and semantics. It is based only on neighbor relations between adjacent brokers and thus can perfectly adapt to heterogeneous environments even with a rather high degree of mobility resulting in a dynamic topology. ASP as a communication approach only provides a system architecture and a routing algorithm with some optional enhancements. In the task of implementing a scenario using the ASP approach a number of problems and challengers arise that need to be overcome to result in efficient communication. In this paper we will outline the most important problems and provide solutions to overcome them.

Therefore the paper is structured as follows. In the following section we present related work. In section 3 we outline the application interfaces and the routing requirements that provide the framework for an ASP implementation. Afterwards we analyze the three phases of ASP and the associated problems and explain, how we solved them. In section 5 we conclude our work and present ideas for future work.

2 Related Work

Publish/subscribe (pub/sub) provides the basis for decoupled communication between an information source and an information sink in time, space and threads [2]. Content-based routing (CBR) [5] implements the pub/sub paradigm in a fully distributed fashion by introducing a network of brokers. Between these brokers subscriptions are distributed in the form of filters that either match a publication or not. Using these filters each broker keeps a local routing table to allow for content-based routing of publications. CBR was enabled to work in mobile ad-hoc networks (MANETs) e.g. by [6][7].

Pub/sub especially for smart environments is provided by MundoCore [8], a modular middleware for the requirements of pervasive computing based on a microkernel design. It supports structured, hierarchical and single-hop strategies for routing resulting in high scalability and adaptability. It allows for channel and content-based subscriptions.

There are many algorithms for on demand routing in MANETs most common probably ad hoc on demand distance vector routing (AODV) [9]. The basic idea of such routing protocols is for a node N_1 to find a communication partner N_2 in a MANET on demand by sending a route request (RREQ). This request is flooded through the MANET until it eventually reaches N_2 which replies with a route reply (RREP). The RREP is send back on the shortest path to N_1 and a communication path is established. The main advantage of on demand routing is that a path is created only on demand and thus, no traffic is induced by nodes that do not communicate. Furthermore through the availability of unique (IP-)addresses and a homogeneous communication protocol (IP) throughout the MANET, these routing algorithms can be highly optimized. If such conditions can not be guaranteed as in heterogeneous ubiquitous environments, these algorithms are not applicable.

3 Requirements

In this section we outline the requirements for the ASP approach induced by its adjacent components which are the applications and the network.

3.1 Source

A source application (source) provides information to other applications by message publication. The availability of these messages is to be distributed by the middleware and the messages have to be communicated towards interested applications.

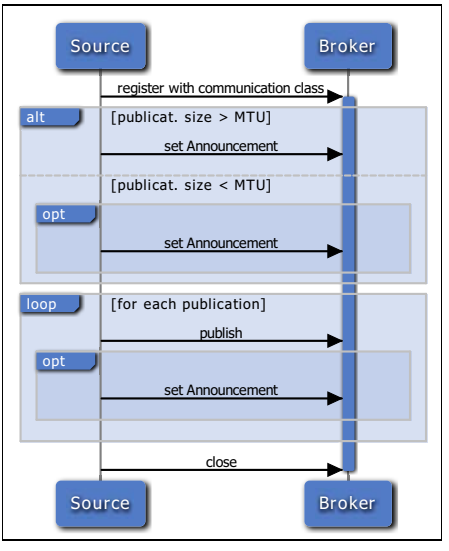

Fig. 1. The interface between source and broker

A source's interface consists of four methods as shown in figure 1. Registration is done stating the communication class[1]. Afterwards an announcement is provided by the source. This is optional if the size of messages is < MTU. At this time the source starts to send messages to the middleware. From that moment on the announcement can be updated anytime by the source. If the source does not send further messages, it closes its registration at the middleware.

3.2 Sink

A sink application (sink) consumes information from other applications by receiving messages from the middleware. Therefore it registers at the middleware with an optional filter to narrow the amount of received announcements. Based on the contents of received announcements it can

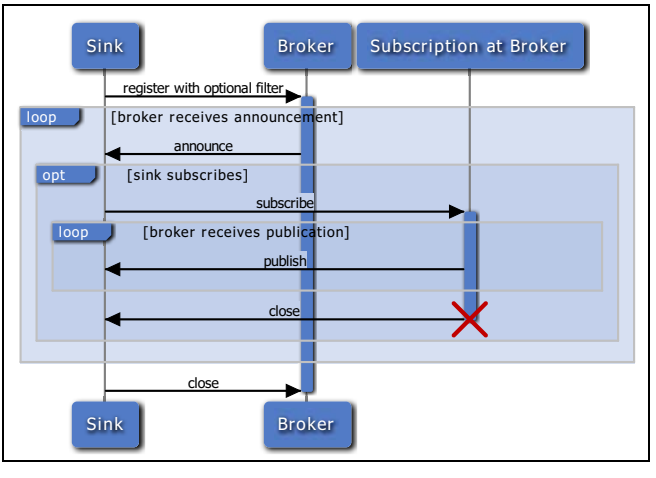

Fig. 2. The interface between sink and broker

[1] The taxonomy given in [4] divides scenarios in four communication classes by message size (one packet (< MTU) vs. fragmentation) and message frequency (one message vs. message stream).

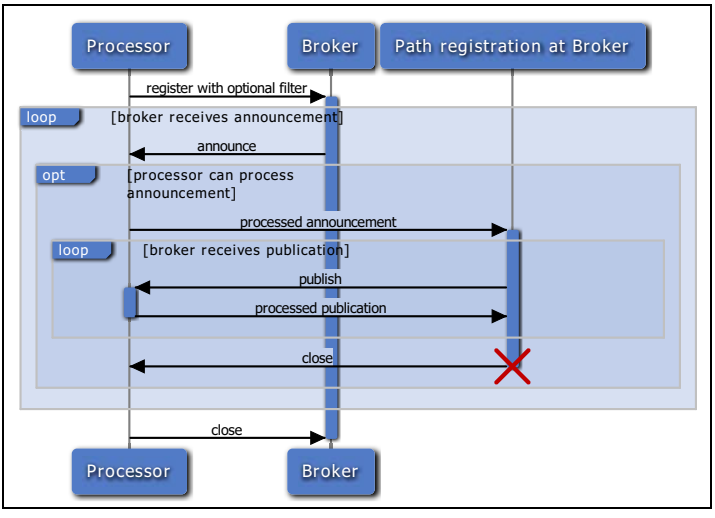

Fig. 3. The interface between processor and broker

subscribe to associated messages. The middleware then delivers all associated messages to this sink. The registration of the sink can be closed completely by the sink itself or partially by the middleware if no more messages are to be expected for the current announcements. The respective interface is shown in figure 2.

3.3 Processor

Processor applications (processors) are needed for message aggregation or processing on the path between source and sink if semantic decoupling is required. Since message processing can be one way in many scenarios, a processor is only required to transform announcements into new announcements and messages into new messages.

A processor's interface is shown in figure 3. A processor registers at the middleware with an optional filter and receives announcements afterwards. If it is able to transform a number of announcements into a new announcement, it sends the new announcement back to the middleware. Additionally, the processor has to provide information about the source announcements and the processing metric. This information is needed by the middleware to optimize paths from source to sink and to avoid communication loops. Later on, if needed, the processor has to transform messages associated to its source announcements into new messages and send these to the middleware. The processor's registration can be closed fully by the processor itself or partially by the middleware if no more messages are to be expected for the current source announcements.

3.4 Efficient Routing

In respect to the network the middleware has to distribute announcements among all brokers and find efficient paths between source brokers and sink brokers for

message delivery. These paths need to involve the right processors if message aggregation or processing is necessary for the communication. The choice of processors needs to be optimized according to the following criteria.

If multiple processing steps are needed, the most efficient combination is to be selected. Efficiency thereby depends on the metric provided by the processors. If multiple equal processors are available, a single one needs to be selected for every path between a source and a sink to avoid unnecessary double processing of messages. If multiple aggregating processors are available, aggregation has to be done as early as possible to optimize network utilization. In the contrary, if processing enlarges the message size, late processing might be useful to again optimize network utilization.

4 ASP-Algorithm

As ASP works in three phases, we analyze each phase for its specific challenges to fulfill the aforementioned requirements.

4.1 Announcement Phase

An announcement is generated by the ASP middleware from the information provided by a source or processor. If generated or received, an announcement is to be distributed to all neighbors except the one this announcement has been received from and to all processors and sinks except for those that provided a filter that does not match the announcement.

Announcement Identifiers. By processing respectively routing through a processor, an announcement's content is changed. To avoid loops where an announcement is processed back and forth again and again and to detect duplicates after similar processing in different processors, announcements need to be compared based on their content. Additionally, announcements need to be referable by later subscriptions, publications and eventually other announcement. Therefore we generate a content based identifier for each announcement upon its generation in a source's or processor's broker using a hash algorithm. Comparison and recognition of an announcement therefore can easily be done by comparing the announcement's identifier.

Based on the implemented scenario, similar processing of an announcement in different processors can lead to slightly different announcements actually representing the same content. One example are arithmetics on floating point values that can lead to different results by rounding. Such announcements result in very different identifiers owing to the hash algorithm. If this can be the case for a scenario, we suggest to extend the source/processor-interface by providing a mask to the ASP middleware to allow the application to unmask the parts of the announcement that should not be used for hash generation.

Announcement Caching. A very efficient way of repairing broken paths in a stream scenario is for the source's broker to start a new announcement after it was informed of the path failure. Since this can happen before the initial

announcement becomes invalid, multiple valid announcements for the same information can be circulating. In combination with announcement caching, where each broker caches all current announcements and forwards them to newly discovered neighbors, the number of announcements to cache and forward rises. In a dynamic scenario path failures can thus increase the network traffic which leads to more path failures and finally to parts of the network being overloaded.

In a scenario where the announcement for a given source and therefore its identifier does not change, this behaviour can be avoided by introducing a sequence number that is incremented on every re-announcement. Thus invalid announcements can be sorted out easily. If however the announcement changes frequently, its identifier changes as well. To recognize invalid announcements each re-announcement can be provided with a list of identifiers of invalidated announcements. This allows receivers of the re-announcement to immediately remove announcements that became invalid. Even the attachment of only the most current invalidated identifier to each re-announcement decreases the occurrence of the aforementioned network overloads to a minimum.

Path Metrics. To evaluate different paths in the process of announcement distribution a path metric is calculated from the link metrics on the announcement's path. These can be metrics of communication links or processing links. This leads to a number of requirements for such a metric:

1. There must be a metric to reflect the transmission costs between two adjacent brokers and a metric to reflect the costs of processing in a processor.
2. A path metric will most likely contain transmission costs and processing costs. Thus, there must be a method to compare or even aggregate them.
3. If processing involves data aggregation, multiple path metrics need to be aggregated as well. This process has to reflect the requirement, that aggregation should be done as early as possible to optimize communication costs.
4. To generate useful path metrics for the publication phase they actually need to reflect the costs of message transmission. Especially the size of a message can be very different from the size of the associated announcement and can largely influence the transmission costs.

In our experimental scenarios we identified time based metrics as very promising to fulfill the first two requirements. The basis for our metric is the expected transmission time (ETT) like in [10]. The ETT is therefore calculated in the network abstraction layer (NAL) [4] by monitoring the packet round-trip time and the number of necessary (re-)transmissions as this approach can easily be generalized for any kind of link layer communication and above. For announcement processing the processing time replaces the ETT as basis for the metric. Inspired by the parameter *willingness* in OLSR [11] we multiply the time with an effort factor to reflect additional criteria like power consumption, monetary costs or network respective CPU utilization. The effort for each connection is dynamically determined by the brokers NAL and the effort for announcement processing is provided by the processor itself.

The path metric P for a path with m links of either announcement forwarding or processing is generated from the connection respective processing metrics M by summation: $P = \sum_{r=0}^{m} M_r$.

This has two main advantages. Firstly an announcement only needs to carry one value to reflect the overall path metric. To append a new step to the path metric its connection or processing metric is added to the announcement's path metric before this step. Secondly, if n announcements A_i are aggregated somewhere on their path, the path metric of the announcement is summed up with the processing metric of their aggregation M to generate the new announcement's (A') path metric: $P_{A'} = M + \sum_{i=1}^{n} P_{A_i}$.

This procedure fulfills the third requirement. If possible, an earlier aggregation of two announcements on the same path will automatically result in a lower path metric for the resulting announcement given that the metric of the early aggregation is about the same as for the later one.

The fourth requirement results in a fundamental problem because the actual size of the message is not necessarily known at the time the announcement is distributed. However, we assume that a processor can estimate the size s' of a message after processing in relation to the size s_i of all n messages before processing by providing the growth factor G with $G = \frac{s'}{\sum_{i=1}^{n} s_i}$.

We propose to add a size factor S to each announcement. The initial value is $S = 1$. On a processing step involving n source announcements, the size factor of the resulting announcement A' is calculated as: $S_{A'} = G \times \sum_{i=1}^{n} S_{A_i}$.

This gives the following general equation for the path metric on each step.

$$P_{A'} = \sum_{i=1}^{n} P_{A_i} + M \times S_{A'} \tag{1}$$

$$= \sum_{i=1}^{n} P_{A_i} + M \times G \times \sum_{i=1}^{n} S_{A_i} \tag{2}$$

However, if a step only involves communication, there is only one source announcement ($n = 1$) and no change of size ($F = 1$) resulting in the following very simple equation.

$$P_{A'} = P_A + M \times S_A \tag{3}$$

The path metric using the size and growth factors should result in a more realistic path metric for the communication of the message afterwards because changes of message size from processing are reflected in the overall path metric of received announcements. Processing steps that enlarge (reduce) a messages size will result in a better overall path metric if performed late (early) in the communication path.

4.2 Subscription Phase

Based on the contents of a received announcement, a sink can decide to receive the associated message or message stream. The delivery of the message or

message stream afterwards is channel based. We call this channel an active path. A path is activated by the sink's broker sending a subscription back to the source's broker.

Path Identifiers. The announcement's identifier is used to identify the contents to be delivered on an active path later. The active path itself needs another unique identifier to allow for multiple active paths for the same content that can be unsubscribed from or repaired independently of each other. Therefore we use a path identifier that is generated randomly by the sink's broker for each subscription it initiates.

In the unlikely event of two sinks generating the same random path identifier for the same contents, the second subscription will overwrite the first one. This results in all further publications being received by the second sink only. However this state will only last until the announcement becomes invalid and another one is initiated. Thereafter new path identifiers are generated by both sink's brokers and another collision is extremely unlikely.

Empty Announcements. The standard flooding algorithm results in the first announcement that is received being forwarded. This does not necessarily have to be the announcement with the best path metric. To find the one path with the best metric one can use an extension we called empty announcements [4]. An empty announcement contains only the announcement's identifier and the path metric and is used to inform a broker of a better path being available after the announcement has already been delivered. The utilization of empty announcements is only useful if very large messages or streams with a very large amount of data are to be delivered. Otherwise the costs of delivering all the empty announcements can easily outreach the gain of the best path.

Using empty announcements results in one announcement and eventually one or more empty announcements for better overall paths being received by a sink's broker. Since that broker never knows if one more empty announcement is to be expected, it would have to send a subscription for the first announcement. For each empty announcement it would have to send a subscription for the new path and afterwards unsubscribe from the old path. This can generate a large amount of unnecessary network traffic.

Our solution to this problem is for the sink's broker to react to the first received announcement with a subscription to assure that one active path exists as soon as possible. To optimize that path later on, the broker collects all empty announcements it receives for a not too small amount of time. After this time has passed it treats the last empty announcement with a better path metric than the ones before as the best path. If such a best path is available, a subscription for that path is initiated. Afterwards the broker unsubscribes from the initial path. This procedure is illustrated in comparison to the trivial approach in figure 4.

This procedure guaranties fast delivery of the first publications to decrease the initial latency. For the expected large amount of later publications the best possible path is used to unload the communication and processing infrastructure. The number of subscription rounds is limited to a maximum of two.

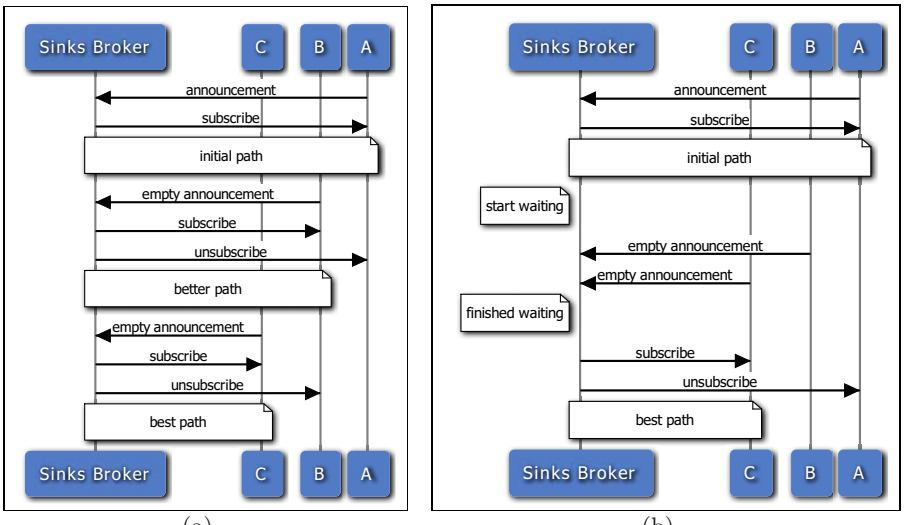

Fig. 4. Case where the sink's broker receives three empty announcements: Instead of adjusting the active path on every empty announcement (a), it waits to collect all empty announcements and adjusts the active path once (b)

4.3 Publication Phase

The purpose of the publication phase is the delivery of messages on every active path. Depending on the communication class, one message can be fragmented into multiple publications. Additionally multiple messages can form a message stream. If a message is fragmented into multiple publications, each publication must be delivered reliably because loosing one fragment would render the whole message useless. If multiple messages are to be transmitted as a stream, it is desirable to deliver as many messages as possible but we assume loosing single messages in a stream not as problematic. Developers of scenarios and applications for ASP have to choose their communication class carefully to reflect this assumption.

Publication Caching. According to the aforementioned behaviour, all publications that belong to a single message, are cached in every broker on the active path. This has two main advantages. On the one hand, this reduces network and CPU utilization. If a new subscription is initiated by a broker that results in an active path which overlaps with an existing active path the publications that have been transmitted on the existing active path do not have to be transmitted on that path again. Instead they can be forwarded immediately by the broker where both active paths stop to overlap. This also reduces the message transmission latency to the aforesaid subscribing broker.

On the other hand, publication caching is a major requirement for time decoupling. Especially if only one eventually fragmented message is to be transmitted

the source might already be disconnected at the time when any subscriptions arrive. Only by caching the publications they can still be delivered towards the subscriber in this case.

5 Conclusions and Future Work

In this paper we analyzed the process of implementing scenarios using the ASP approach for content based decoupled communication based on neighbor relations. After specifying the necessary requirements for application interfaces and the behaviour of the middleware we identified six circumstances where the ASP approach could lead to problems and presented solutions how to solve these problems:

- different announcement identifiers for the same content,
- multiplication of invalid announcements through announcement caching,
- path metrics for optimal paths,
- path identifiers to keep active paths independent,
- a re-subscription algorithm to reduce subscription overhead if empty announcements are used, and
- publication caching to assure time decoupling.

In the following sections we present advantages and disadvantages of implementing a scenario in heterogeneous ad-hoc environments using the ASP approach.

5.1 Advantages

The only precondition for the network topology of a scenario to be implemented using ASP is that a bidirectional link-layer communication of any kind between neighboring nodes is available in a way that the nodes and communication paths form a connected graph. The degree of heterogeneity does not matter. Globally unique network addresses are not necessary.

Because of its decoupled nature, the ASP approach is very flexible in dynamic environments and therefore allows for mobility and ad-hoc topologies involving mobile components.

If a scenario is implemented using ASP, all applications - sources, sinks and processors - are fully decoupled from each other and the network in terms of location, availability and semantics. This enables independent development of applications. Another advantage is the resulting completely distributed infrastructure. Applications can be added, removed and even used in parallel at any time.

5.2 Shortcomings

Even though using ASP has many advantages we do not want to forget to mention its shortcomings. The most important one is probably that due to the utilization of flooding algorithms, ASP does not scale to arbitrary large networks. Boundaries for the distribution of announcements are necessary in a physical,

an algorithmic and a logical way. Physical boundaries can be environments of limited size like smart or ubiquitous environments. Algorithmic measures are e.g. a time to live or context based means to limit the distribution of an announcement. With logical boundaries we refer to the fact that especially if algorithmic measures are taken the developer needs to understand that the availability of information is not distributed and thus known everywhere anymore.

Another disadvantage owing to the nature of the pub/sub paradigm might be that ASP is unidirectional. Content is delivered from the source to the sink only. In most scenarios especially for content based communication this is not a problem. However there might be the necessity for bidirectional communication. If bidirectionality is limited to a confirmation of reception the ASP approach could probably be extended without much problems to provide such a confirmation. But this would destroy the very idea of decoupled communication in time and especially in threads. Therefore we did not propose such an extension ourselves.

5.3 Future Work

For future research we plan to investigate three open questions: Can the ASP approach be extended to not only allow the implementation of single scenarios but to provide a toolkit for ASP based application developing?

Can the ASP approach be extended to provide more quality of service features for applications? Right now ASP allows for applications to provide a communication class to optimize communication which means it can select between optimization for small vs. large messages and optimization for single messages vs. a message stream. Features like reliability vs. load-efficiency or real time demands for applications could be very interesting as well.

Can ASP be extended to provide efficient bidirectional communication in heterogeneous ad-hoc environments? The availability of such bidirectional communication could allow for e.g. content based service announcement with subsequent channel based service utilization.

Acknowledgement

Henry Ristau is supported by a grant of the German National Research Foundation (DFG), Graduate School 1324 (MuSAMA).

References

1. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Comput. Surv. 35(2), 114–131 (2003)
2. Aldred, L., van der Aalst, W.M., Dumas, M., ter Hofstede, A.H.: On the notion of coupling in communication middleware. In: Meersman, R., Tari, Z. (eds.) OTM 2005. LNCS, vol. 3761, pp. 1015–1033. Springer, Heidelberg (2005)
3. Aldred, L., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Dimensions of coupling in middleware. Concurrency and Computation: Practice and Experience (February 2009)

4. Ristau, H.: Announcement/subscription/publication: Message based communication for heterogeneous mobile environments. In: Mobile Wireless Middleware, Operating Systems, and Applications, Mobilware 2009, Berlin, Germany (2009)
5. Carzaniga, A., Wolf, A.L.: Content-based networking: A new communication infrastructure. In: König-Ries, B., et al. (eds.) IMWS 2001. LNCS, vol. 2538, pp. 59–68. Springer, Heidelberg (2002)
6. Baldoni, R., Beraldi, R., Cugola, G., Migliavacca, M., Querzoni, L.: Structure-less content-based routing in mobile ad hoc networks. In: Proceedings of International Conference on Pervasive Services, ICPS 2005, July 11-14, pp. 37–46 (2005)
7. Petrovic, M., Muthusamy, V., Jacobsen, H.A.: Content-based routing in mobile ad hoc networks. In: Mobile and Ubiquitous Systems: Networking and Services, MobiQuitous 2005, pp. 45–55 (2005)
8. Aitenbichler, E., Kangasharju, J., Muhlhauser, M.: Mundocore: A light-weight infrastructure for pervasive computing. Pervasive and Mobile Computing (2007)
9. Perkins, C., Belding-Royer, E., Das, S.: Request for Comments: 3561 - Ad hoc On-Demand Distance Vector (AODV) Routing. RFC (July 2003)
10. Draves, R., Padhye, J., Zill, B.: Routing in multi-radio, multi-hop wireless mesh networks. In: MobiCom 2004: Proceedings of the 10th annual international conference on Mobile computing and networking, New York, NY, USA, pp. 114–128 (2004)
11. Clausen, T., Jacquet, P.: Request for Comments: 3626 - Optimized Link State Routing Protocol (OLSR). RFC (October 2003)