Announcement/Subscription/Publication: Message Based Communication for Heterogeneous Mobile Environments

Henry Ristau*

Chair for Information and Communication Services
University of Rostock
Albert-Einstein-Str. 21
18059 Rostock
Germany
henry.ristau@uni-rostock.de

Summary. Many tasks in smart environments can be implemented using message based communication paradigms that decouple applications in time, space, synchronization and semantics. Current solutions for decoupled message based communication either do not support message processing and thus semantic decoupling or rely on clearly defined network structures. In this paper we present ASP, a novel concept for such communication that can directly operate on neighbor relations between brokers and does not rely on a homogeneous addressing scheme or anymore than simple link layer communication. We show by simulation that ASP performs well in a heterogeneous scenario with mobile nodes and decreases network or processor load significantly compared to message flooding.

Keywords: Heterogeneous Mobile Environments, Publish/Subscribe, Distributed Event-Based Systems, Content-Based Routing, Flooding.

1 Introduction

A growing application area for heterogeneous wireless networks are smart environments. They emerge from the cooperation of many different devices from tiny sensors to powerful computers. This cooperation targets at a specific objective which in smart environments usually is user assistance. While communication between applications on all of these devices is a basis for cooperation, heterogeneity and mobility induce major problems here. Many devices can communicate using multiple communication technologies, the technologies themselves however are often incompatible. Homogeneous addressing schemes are not guaranteed to exist throughout a smart environment. The network topology is constantly changing

* This research was supported by a grant of the German National Research Foundation (DFG), Graduate School 1424, Multimodal Smart Appliance Ensembles for Mobile Applications (MuSAMA).

C. Giannelli (Ed.): Mobilware, LNICST 0007, pp. 295–308, 2009.

through entering, leaving and mobile devices. Thus, to keep applications simple it is essential that communication in a smart environment is as transparent from the applications as possible.

Another problem in smart environments is the heterogeneity of devices and applications themselves. Applications are supposed to communicate even if they use different document or data formats. Data from multiple devices might have to be aggregated while it is communicated towards its sink. Again from the perspective of an application developer this data processing should be as transparent as possible.

An example is the communication between a number of sensors and a PDA in a heterogeneous environment. The applications on the sensors should only measure and provide the temperature. Where, when and in which unit of measurement this temperature is needed by other devices in the network should be of no interest to these applications. On the other hand, the application on the PDA should only display the room temperature. It should not be concerned about where the temperature is measured, which units are used, and how many sensors are available. The data communication and processing between the sensors and the PDA should be efficient and reliable. However it is not important how many communication technologies are involved or how many systems are able to do the processing, where these systems are located and if they are available.

In this paper we present Announcement/Subscription/Publication (ASP), a concept to achieve message based communication between applications decoupled of time, space, synchronization and semantics in heterogeneous mobile environments. ASP allows an application to publish or receive and subscribe to messages transparently with respect to the network infrastructure, the availability and location of communication partners, asynchronously of any broker or communication partner and transparently with respect to the semantic capabilities of any communication partner. To achieve the transformation and processing of messages, an application can register as processor. Such a processor application only has to process messages according to its capabilities with the same transparency as described before. It does not need to be able to describe its processing abilities - of course it helps if it can somehow subscribe to its preferred input messages.

The remainder of this paper is structured as follows: In the next section we explain which kind of scenarios for message based communication we tackle in this paper. In section 3 we present related work and give an elaborate problem statement followed by a detailed description of the concept of ASP in section 4. In section 5 we present the results of an evaluation of our concept and conclude the paper in section 6.

2 Application Scenarios

Scenarios for message based communication in smart environments can be characterized by message size and publication frequency. Small messages can be transmitted from one device to another one completely before they are forwarded

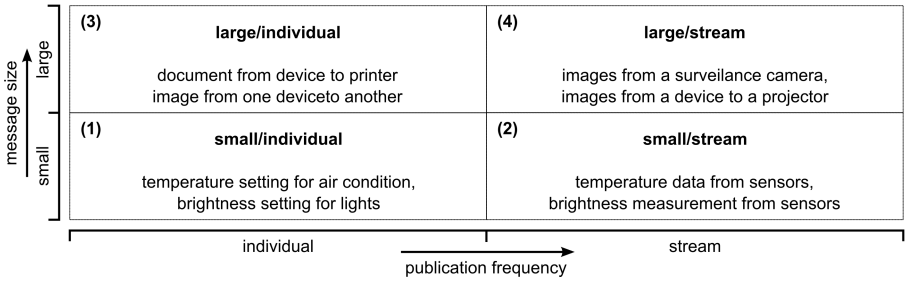

Fig. 1. Scenarios for message based communication divided into four groups according to message size and publication frequency

to a third device without a significant loss of performance. Preferably they fit into one network packet. Large messages need to be fragmented and should only be received completely if necessary, e.g. for processing to avoid high transmission latencies. Individual messages are transmitted only once or so infrequent that it is not efficient to keep a routing path for the next message. Streams of messages however are transmitted frequently on the same paths. This division results in four groups labeled with some examples for each group in figure 1.

In the first group flooding algorithms are the method of choice sending the individual message to all possible target locations. This results in the message being delivered to every available sink. However in any other group more than just one data packet is delivered and flooding algorithms can easily overload parts of the communication infrastructure or data processors with less performance. In this paper we concentrate on scenarios of the second group and conclude some means to extend our ideas for scenarios of groups three and four which we however do not yet evaluate.

3 Related Work and Problem Description

Publish/subscribe is the communication paradigm of choice for decoupling of message source (publisher) and sink (subscriber) in time, space and synchronization [6] using a central broker to register the interest of subscribers and subsequently forward publications to them. By content-based routing (CBR) [4] it is possible to replace the broker as former central element by a network of distributed brokers and decouple the publish/subscribe system from the underlying communication technologies at the same time.

Apart from subscribing to a message subject or channels, CBR allows subscriptions to the content of a message, which can be for example values of a tuple. The routing tables for CBR are initialized based on interest of subscribers that is disseminated through the network using flooding techniques. Depending on the resulting routing tables, publications can be routed from publisher to subscriber directly in a multipath fashion [3]. The distribution of interest was optimized through the usage of beacons. This increases the fault-tolerance of the

routing protocol to work in mobile environments like mobile ad hoc networks (MANETs) [2][14].

Publish/subscribe especially for smart environments is provided by MundoCore [1], 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.

The processing of messages however is not integrated into or observed by the publish/subscribe middleware. In [15] we analyzed two approaches of integrating processors into a publish/subscribe middleware. The first one, mapping the processor to a sink for its input and a source for its output messages, induces much unnecessary processing and communication if multiple processors for the same operation are available. The second approach, the extension of CBR to allow dissemination of interest through processors, leads to very complex processors because apart from processing message type m to message type m' they need to be capable of processing a subscription for message type m' into a subscription for message type m which could even be impossible depending on the type of processing.

Two very specialised approaches for message processing in communication are composite event detection and event stream processing (ESP). The former one denotes the composition of primitive events to monitor the state of a distributed system and notify sinks of the detection of composite events. One such system is GEM [12]. ESP systems like Borealis [7] or Cayuga [5] are able to execute a continuous query over a stream of events delivering the queries result to sinks. Both systems have in common that the sink has to provide the processing instructions in the form of a script or a query and thus is not decoupled from the distributed system in terms of semantics.

A system for semantically decoupled communication in sensor networks is presented in [17]. Data from different sensor networks is collected in an IP-based overlay network where the semantic decoupling is done using ontologies. The resulting information is provided to connected sinks. The system is based on a very strictly defined topology and thus not applicable for heterogeneous mobile networks as targeted by this paper.

4 Contribution

The ASP concept consists of a message routing algorithm based on a given system architecture. We start by introducing the system architecture. Afterwards we describe the three phases of the routing algorithm, Announcement, Subscription and Publication. At the end of this section we identify the requirements to implement a scenario using the ASP concept.

4.1 The System Architecture

To decouple the applications transparently from the network and all other applications we base our approach on a system architecture with one broker on

each participating node. All client applications are connected locally and other brokers are connected through the network layer. We do not consider remote applications on nodes without broker because it is no problem to implement very slim brokers for nodes with less performance. Each broker keeps a cache of virtual neighbors, which can be applications - source, sink and processor - or neighboring brokers that can be reached with one hop through any communication technology as shown in figure 2.

Applications Interfaces. Source applications register at the broker and have two main methods available. They can register a special descriptive announcement message (see section 4.2) if they wish to and they can publish messages.

A sink application can register using an optional filter. If a filter is provided the sink receives only announcements matching the given filter, otherwise they receive every announcement received by their broker. After receiving an announcement, a sink application can decide to subscribe to that announcement and will receive further publications from that source.

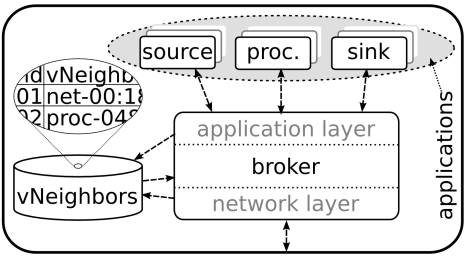

Fig. 2. The system architecture for ASP features one broker per node to transparently decouple applications from each other and the network topology

Processor applications register just like sinks with an optional filter and receive announcements. If they are able to process any of these announcements, the result is to be submitted back to their broker along with identifications of the processed source announcements and a metric information denoting the complexity of the processing. If the processor application is part of an active path later, further publications are sent to it for processing.

Network Abstraction Layer (NAL). The NAL decouples the routing algorithm from any underlying communication technology. Its tasks are neighbor discovery, neighbor cache updates and reliable message delivery. Brokers in communication range have to be detected and inserted into the neighbor cache. Their information in the cache, especially their connection metric, has to be updated if it changes. Each message from the broker has to be delivered to the neighbor or if it can not be delivered, the neighbor has to be removed from the neighbor cache because it is not reachable anymore. For any of these actions, the broker has to be notified.

Since the algorithm relies on communication links to its direct neighbors only and the NAL provides neighbor discovery and reliable message delivery, ASP can work directly on the link layer if necessary and can easily be implemented for any communication technology by adding the appropriate NAL.

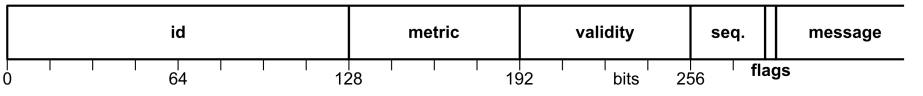

Fig. 3. An example announcement with all necessary data fields

Terminology. Subsequently we denote a source application's broker as "source", a sink application's broker as "sink" and a processor application's broker as "processor". A virtual neighbor of a broker, which can either be an application or a neighboring broker, is meant by the term "neighbor" if not specified.

4.2 The Announcement Phase

When a source application publishes a message, the source disseminates the availability of this message among all brokers by sending an announcement.

Contents. Figure 3 shows a minimal example for an announcement. The id is generated together with the announcement by the source or processor using a defined hash algorithm on the message's content right after the announcement is created. It is used as content-based reference for the announcement and for duplication and loop detection in the announcement's distribution.

The metric is updated by every broker before the announcement is sent to a neighbor and represents the path metric between the source and the receiving broker. The validity value is initiated by the source and represents the time the announcement is valid. If an announcement is not prolonged before, all references to it can be removed from a broker's cache. To recognize an identical announcement with the task to prolong an older one, the sequence number has to be incremented. The flags are needed in an extension of the algorithm to distinguish individual mode from stream mode.

The message is the descriptive announcement message registered or the next message published by a source application, or results from processing by a processor application. The requirement for a message in an announcement is that it has to be small enough for the announcement to fit into one packet the NAL can send out. The distribution algorithm itself is not meant to support announcement fragmentation while packet fragmentation can be implemented in the NAL if needed.

Distribution. The target of announcement distribution is for every broker to receive the announcement and store the neighbor it was received from with the lowest path metric as best announcer. Hence, if a sink application is interested in this announcement, its broker can subscribe to this optimal path of distribution. Therefore, an extended flooding algorithm is utilized.

The full announcement is sent to every neighboring broker, except for the sender if that was a neighboring broker. Furthermore it is distributed to every

Fig. 4. An example subscription with all necessary data fields

processor and sink application except if they have registered using a filter and the filter does not fit the announcement. If an announcement is re-received with a better path metric, the message is stripped off and the resulting short announcement is forwarded according to the same rules as above to signal the better path to the following brokers. Of course a short announcement is not forwarded to sink applications because they can gain no new information from it and its processing should only be simulated by the broker knowing the outcome and metric from the preceding full announcement because the processor application could not do any processing without the message.

4.3 The Subscription Phase

The subscription represents the control data in the ASP concept. It is used for brokers to signal a subscription for a given path turning that path into an active path. Other purposes are the removal of an active path or the indication of a broken path towards the source for re-announcement.

Contents. Figure 4 shows the necessary fields for a subscription. The id is used as reference for the associated announcement. The subscriberId is generated randomly by the sink to represent that broker as the subscriber. It is needed to distinguish between different subscriptions for the same announcement to allow multiple sinks. Two identical subscriberIds would eventually result in one sink not receiving its publications for this announcement period but the probability of this event is sufficiently low. The metric is also set by the sink to the metric of the best announcement received so far to allow path optimization if a better announcement is received after a subscription has been generated.

The type signals the purpose of a subscription. So far, possible types are subscribe, unsubscribe, and broken path. The sequence value is used to allow for publication caching which is not yet necessary for small/stream scenarios as described in section 2.

Distribution. If a new subscription or one with a better metric value is received from a neighbor, that neighbor is marked as active path for the provided id and subscriberId. Afterwards the subscription is forwarded to the best announcer if it is a neighboring broker. If the best announcer is a processor application, one subscription for every stored source announcement id is generated and forwarded as described before. The neighbor a subscription is sent to is stored by the broker.

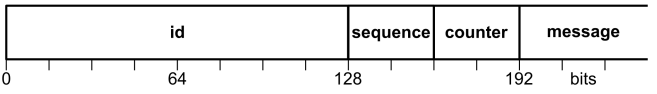

Fig. 5. An example publication with all necessary data fields

If the best announcer changes after any subscription has been sent out, a new subscription is sent out according to the above rules and an unsubscribe subscription is sent to the target of the previous subscription to remove the obsolete active path. If an unsubscribe subscription is received that matches an active path with its subscriberId and metric, the active path is removed and the unsubscribe subscription is forwarded to the former subscriptions target.

If an active path is detected to be not available anymore for any reason, a path broken subscription is generated and forwarded according to the same rules as above. If a source receives such a path broken subscription, it prepares to extend the next publication to an announcement so a new path can be found if available.

4.4 The Publication Phase

Following an announcement, the source will send out all published messages as publications until the next announcement is necessary due to announcement expiration or path disintegration. A minimal publication is depicted by figure 5. The id is used as reference for the associated announcement and subscriptions. The sequence number is for fragmentation which is not necessary for small/stream scenarios. The counter is used to recognize and remove unwanted retransmissions.

A publication is forwarded on all known active paths just once. Hence, every subscribed broker receives the publication and forwards it towards the subscribed sink applications.

4.5 Requirements for a Scenario

ASP is a communication concept which can be implemented for a given scenario or a full middleware if desired. A number of requirements have to be met by the scenario which we present in the following.

Announcements. If published messages are too large to fit into an announcement (cf. 4.2) or descriptive announcements are needed for other reasons, a means of describing messages is needed. Examples are meta-data as in document headers or advertisements as in many publish/subscribe systems. Processor applications receiving a descriptive announcement must be able to decide whether they can process the following publications.

Furthermore a hash algorithm is necessary to generate a sufficient unique id from a published message or descriptive announcement. Eventually the accuracy of floating point representation has to be limited before hashing to avoid multiple instances of the same messages if processing by different processing applications can lead to different rounding.

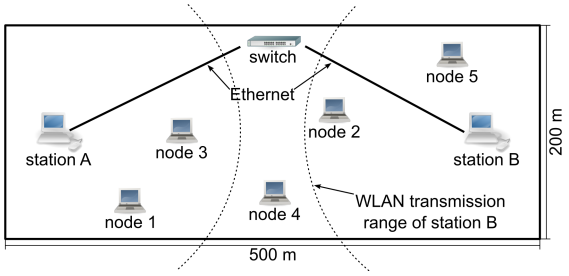

Fig. 6. Simulation scenario: station A and B are connected by Ethernet while all nodes and stations are connected by wireless LAN with the given estimated transmission range. The nodes move inside the 200m x 500m playground area.

Metrics. A metric is needed that can describe communication and processing to result in optimal paths for message publication. We prefer time based metrics because time can equally be measured for processing and communication and it can be totalled to result in a path metric even if processing leads to data aggregation.

System Size. Since flooding does not scale for very large topologies a system boundary is necessary. This can either be a limited environment like a smart environment we are targeting, a limited number of systems that implement a broker or an additional measure like for example a hop count or a maximum metric for the flooding of announcements.

5 Evaluation

Based on the ASP concept we implemented a scenario with source, processor and sink applications, a broker implementation and a NAL for 802.2 logical link control (LLC) [9] for simulation. Since 802.2 LLC is the link control layer for 802.3 Ethernet [10] and 802.11 wireless LAN [8] our scenario includes stationary devices as well as mobile nodes. We evaluated the completeness of message delivery and the load induced by communication.

5.1 Simulation Scenario and Methodology

Figure 6 shows the simulation scenario. It consists of two stationary nodes, station A and B, connected by Ethernet and five mobile nodes. All nodes are connected by wireless LAN in ad-hoc mode according to 802.11b [8] with 11MB/s, a maximum transmission power of 1 mW, and a limit of 3 retransmission. The mobile nodes are moving according to the Random Waypoint Mobility model [11] on the full playground with a random velocity of 1 to 5 m/s and random pause lengths of 1 to 5 seconds (both uniform distribution).

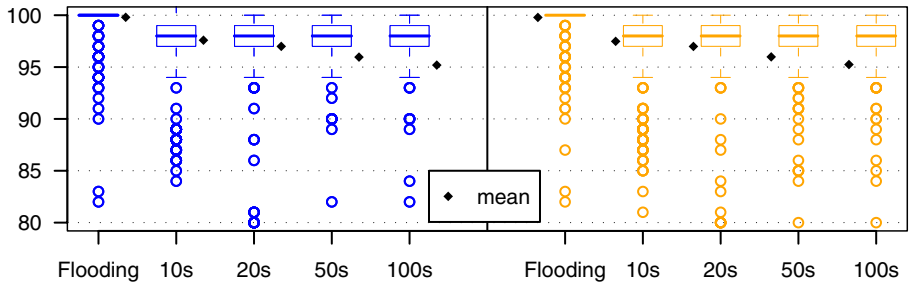

Fig. 7. Box-plots showing the number of messages received by sink 0 (left) and sink 1 (right) using Flooding compared to ASP with different validity settings (10s, 20s, 50s, and 100s) from 100 messages sent by the source

The scenario is a small/stream scenario with one temperature source application on node 1 that generates 100 temperature values in degrees Celsius, one every second. One processing application is on station A (Celsius to Fahrenheit) and one on station B (Celsius to Kelvin). Two temperature sinks are on node 2 (Fahrenheit) and node 3 (Kelvin).

The scenario was implemented for the OMNeT++ discrete event simulator [16] version 3.3 and the INET Framework version 20061020 [13]. The experiment was run 1000 times with different seeds, leading to different starting positions for the mobile nodes and different movement patterns.

5.2 Completeness

We define completeness as the number of messages that are delivered to a given sink application relative to the number of messages that could have been delivered if all messages published by any source application would have been processed by all possible processor applications and delivered to that sink application in the time of observation. Thereby the sink and any processor application only count as available until the time they stop the registration at their local broker for the last time in the observation interval.

In a mobile scenario a completeness of 1 as the theoretical maximum is not always achievable because the availability of a path through the network topology is not taken into account in the definition of completeness. Therefore we compare the completeness of the ASP concept with the flooding of every message to all brokers. A higher completeness than flooding is possible if caching is used which however is not implemented in our small/stream scenario because the loss of single messages is often not a problem in such scenarios.

Results. Figure 7 shows the number of messages received by sink 0, the sink application on node 2, and sink 1 on node 3 for flooding and ASP with differing announcement validity. For flooding while most runs result in all 100 messages

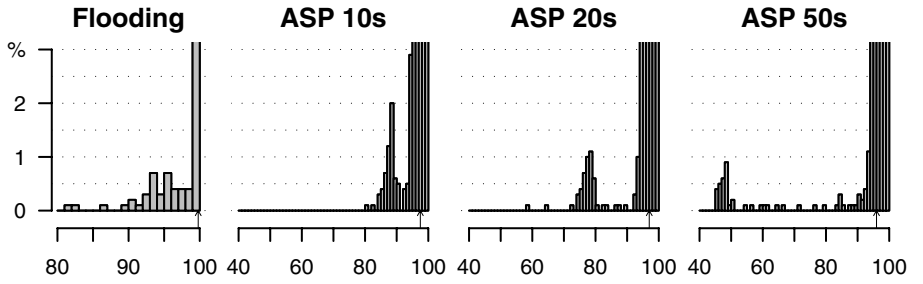

Fig. 8. Histograms showing the number of messages received by sink 1 on the x-axis and the percentage of experiment with this result on the y-axis. The small arrow in the bottom of each histogram shows the mean value.

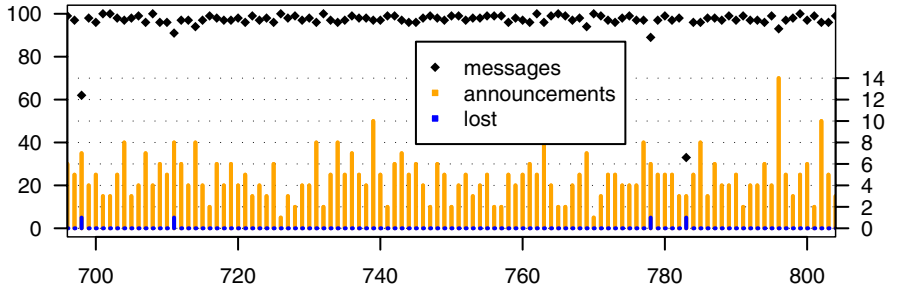

Fig. 9. The number of announcements initiated by the source, the number of announcements not delivered to the sink and the number of messages received by sink 1 for the given runs of the experiment using ASP with a validity of 100s

being delivered, a number of topology states lead to messages being lost. The mean value with 99.8 for both sinks is very high. For ASP we recognized that for most runs less than 6 messages are lost. However there are some outliers leading to a drop of the mean value down to about 95 messages for a validity of 100 seconds.

The histograms for sink 1 in figure 8 show that in most runs, only few messages are not transmitted. There is also a significant fraction of runs where x or a few more messages are missed with $x = t/1s$, with t being the announcement validity and $1s$ being the time between every 2 messages.

Discussion. When an active path breaks in publication phase the publication is not retransmitted and lost. This leads to a single message being lost and can be recognized in figure 9 by more than one announcements being initiated. Without path failures in that experiment one announcement should suffice which can be seen in run 726 and 770 because the validity for ASP equals the duration of the experiment.

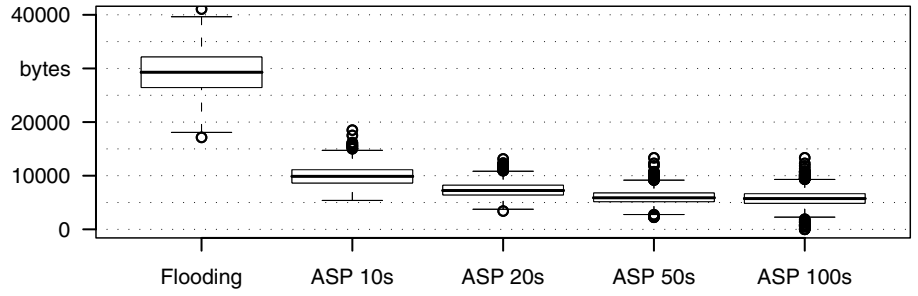

Fig. 10. Box-plots showing the average number of network transmissions sent by each node using Flooding and ASP with different validity settings

The histogram of the flooding algorithm in figure 8 shows that there are phases when no path is available at all. If this happens in announcement phase, all publications are lost up to the next announcement. These are mostly $x = t/1s$ messages as observed before, or eventually less messages if the experiment stops or a path to another sink is lost earlier. This behaviour can be observed in more detail in figure 9 for runs 698, 711, 778, and 783.

5.3 Network Load

To represent network load we measure the number of bytes sent out by the NAL of every node for the purpose of message transmission. We do not count data sent for management purposes like neighbor discovery or heartbeats because we want to compare the broker's dissemination algorithms and not the NAL implementations. Again we compare to simple flooding as the more flexible algorithm.

Results. Using the flooding algorithm, every node sends an average of 30 kbyte of algorithm messages throughout the experiment as shown in figure 10. Using ASP with a validity of 10 seconds this is reduced to 10 kbyte and is further reduced with a longer validity down to about 5 kbyte for 100 seconds.

Discussion. For announcement delivery we use an extended flooding algorithm that eventually generates more load than flooding of a single message depending on the metric and the topology of the scenario. Extending the announcement validity does not lead to a linear decrease of initiated announcements because of path failures. Furthermore subscriptions also need to be delivered. Therefore, a network load inversely proportional to the announcement validity cannot be expected. However in our scenario ASP reduces the network load significantly compared to simple flooding.

6 Conclusion

In this paper we presented ASP, a concept for message based communication in heterogeneous mobile networks decoupling source, sink and processor applications as well as brokers in time, space and synchronization. By transparent processing message delivery from source to sink applications is decoupled in semantics as well.

6.1 Benefits

The algorithm significantly decreases network traffic and processor load by using optimal paths in subscription and publication phase. Since control messages are only delivered in subscription phase on optimal paths, their ratio in overall traffic is very low.

The ASP concept relies on neighbor relations between brokers only and thus, it does not need a consistent addressing scheme throughout the network topology. Through separation of the NAL which can directly operate on link layer, it can easily be implemented for any available communication technology.

In our simulation we could show that even though the nodes were mobile in the ad hoc network and the neighbor relations between nodes changed frequently, ASP was able to adapt. Only few messages where lost compared to message flooding in most runs while network load was reduced significantly.

6.2 Shortcomings

If an active path breaks in publication phase, a new announcement is initiated. This leads to more load on the network and processors. Since the number of damaged communication paths very probably rises with a large scenario or more mobility, this is still an issue.

If there is no connection possible between a source and a sink, the sink application will not receive any message throughout the entire publication phase. This is also the case if a new sink applications enters the system after the announcement phase. This leads to a trade-off between flexibility and resource usage. A shorter announcement validity leads to a faster integration of "new" sink applications while a longer announcement validity leads to less network and processor load.

6.3 Future Work

For future work more scenarios need to be implemented and analyzed by simulation or observation. Especially large/individual or large/stream scenarios pose different requirements on the ASP concept because the loss of a single message will become important when it leads to the loss of one or even the only larger message to be delivered. Publication caching is one important step to fulfill these requirements.

Another task is the elimination or at least optimization of the aforementioned shortcomings. Communication and processing load can be reduced if a broken path can be repaired without the initiation of a new announcement by exploiting multipath propagation characteristics of the utilized flooding algorithm. Furthermore announcements can be cached to allow for faster integration of new sink

applications even with high announcement validity. These enhancements will also help ASP in dealing with large/individual and large/stream scenarios.

References

1. Aitenbichler, E., Kangasharju, J., Muhlhauser, M.: Mundocore: A light-weight infrastructure for pervasive computing. Pervasive and Mobile Computing 3(4), 332–361 (2007)
2. Baldoni, R., Beraldi, R., Cugola, G., Migliavacca, M., Querzoni, L.: Structure-less content-based routing in mobile ad hoc networks. In: International Conference on Pervasive Services, 2005. ICPS 2005. Proceedings, July 11-14, pp. 37–46 (2005)
3. Carzaniga, A., Rutherford, M.J., Wolf, A.L.: A routing scheme for content-based networking. In: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, March 7-11, vol. 2, pp. 918–928 (2004)
4. Carzaniga, A., Wolf, A.L.: Content-based networking: A new communication infrastructure. In: König-Ries, B., Makki, K., Makki, S.A.M., Pissinou, N., Scheuermann, P. (eds.) IMWS 2001. LNCS, vol. 2538, pp. 59–68. Springer, Heidelberg (2002)
5. 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)
6. 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)
7. Hwang, J.-H., Cetintemel, U., Zdonik, S.: Fast and reliable stream processing over wide area networks. In: Cetintemel, U. (ed.) Proc. IEEE 23rd International Conference on Data Engineering Workshop, pp. 604–613 (2007)
8. IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999) (December 2007)
9. ISO Std 8802-2: 1998; IEEE Std 802.2-1998 (December 1989)
10. IEEE Std 802.3-2005 (Revision of IEEE Std 802.3-2002 including all approved amendments). Section 1 - 5 (2005)
11. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. In: Mobile Computing, pp. 153–181 (1996)
12. Mansouri-Samani, M., Sloman, M.: GEM: A Generalised Event Monitoring Language for Distributed Systems. Distributed Systems Engineering 4(2), 96–108 (1997), `http://www.iop.org/EJ/article/0967-1846/4/2/004/ds7204.pdf`
13. OMNeT++ Community Site (November 17, 2008), `www.omnetpp.org`
14. Petrovic, M., Muthusamy, V., Jacobsen, H.-A.: Content-based routing in mobile ad hoc networks. In: The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005, July 17-21, pp. 45–55 (2005)
15. Ristau, H.: Publish/process/subscribe: Message based communication for smart environments. In: 2008 IET 4th International Conference on Intelligent Environments (July 2008)
16. Varga, A.: The OMNET++ discrete event simulation system. In: Proceedings of the European Simulation Multiconference, pp. 319–324 (June 2001)
17. Wun, A., Petrovi, M., Jacobsen, H.-A.: A system for semantic data fusion in sensor networks. In: Proceedings of the 2007 inaugural international conference on Distributed event-based systems, pp. 75–79. ACM Press, New York (2007)