Aspect-Oriented Space Containers for Efficient Publish/Subscribe Scenarios in Intelligent Transportation Systems

Eva Kühn[1], Richard Mordinyi[1], Laszlo Keszthelyi[1], Christian Schreiber[1], Sandford Bessler[2], and Slobodanka Tomic[2]

[1] Space-based Computing Group
Vienna University of Technology
1040 Vienna, Austria
{eva,rm,lk,cs}@complang.tuwien.ac.at
[2] Telecommunications Research Centre Vienna
Donau-City 1, A-1210 Vienna, Austria
{bessler,tomic}@ftw.at

Abstract. The publish/subscribe paradigm is a common concept for delivering events from information producers to consumers in a decoupled manner. Some approaches allow the transportation of events even to mobile subscribers in a dynamic network infrastructure. Additionally, durable subscriptions are guaranteed exactly-once message delivery, despite periods of disconnection from the system.

However, in some application areas, like in the safety-critical telematics, durable delivery of events is not sufficient enough. Short network connectivity time and small bandwidth limit the number and size of events to be transmitted hence relevant information needed for safety-critical decision making may not be timely delivered.

In this paper we propose the integration of publish/ subscribe systems and Aspect-oriented Space Containers (ASC) distributed via Distributed Hash Tables (DHT) in the network. The approach allows storage, manipulation, pre-processing, and prioritization of messages sent to mobile peers during bursts of connectivity.

The benefits of the proposed approach are a) less complex application logic due to the processing capabilities of Space Containers, and b) increased efficiency due to delivery of essential messages only aggregated and processed while mobile peers are not connected.

We describe the architecture of the proposed approach, explain its benefits by means of an industry use case, and show preliminary evaluation results.

1 Introduction

The publish/subscribe (pub/sub) paradigm [1] is a common and largely recognized concept for delivering messages (events) in an anonymous decoupled fashion from publishers to peers subscribed for a topic or for the content of a message.

R. Meersman, T. Dillon, P. Herrero (Eds.): OTM 2009, Part I, LNCS 5870, pp. 432–448, 2009.

Current implementations of and research in notification systems are mostly focusing on an effective and large-scale dissemination [2], [3], [4] of huge quantity of information from publishers to subscribers in a fault-tolerant manner, how to improve the semantical quality or the expressiveness of subscriptions [5], [6], how to ensure durability or the correct order of messages [7], [8]. Other pub/sub approaches deal with these issues as well but assume additionally that peers are mobile [9] or the entire network is completely dynamic [10], [11].

In some application areas the durable delivery (in other words the guaranteed delivery with "exactly once" semantics) of subscribed messages is essential. However, there are application domains, like safety-critical telematics, in which this kind of reliability for subscribed events may be considered a precondition for operation, due to jurisdictional reasons, but is not adequate at all. Among others, a durable notification service has to store any events a peer has subscribed for while the subscriber is off-line. Once the peer is reachable again, the saved events have to be delivered to the associated subscriber. This means that the peer would receive a large amount of data that it has to process locally in order to extract relevant information. However, in scenarios from Intelligent Transportation Systems (ITS), mobile peers (vehicles) have only a few seconds of connectivity and very limited bandwidth [12], [13]. This may cause several problems: the reconnecting peer should receive all stored events which may have very different importance for the user or be even stale, but due to the limited bandwidth and connectivity window only a very few messages can be forwarded to the peer creating a kind of back-pressure in the system. Furthermore, due to the small connectivity window, there is a possibility that essential information, such as safety-critical ghost driver warnings, cannot be transmitted to the peer. If such messages are not forwarded to the peer on time humans lives may be jeopardized. Therefore, the safety risk grows with the amount of irrelevant or even incorrect information delivered instead of important life-saving information.

In [14] we described a customizable storage component, called Space Container, for efficient storage and retrieval of structured data. In [15], [16], [17], [18] we presented the SABRON approach on how to distribute and replicate such Space Containers by means of Distributed Hash Tables (DHT) [19] to efficiently store and retrieve structured, spatial-temporal data in a fault-tolerant manner. In this paper we propose the concept of Aspect-oriented Space Container (ASC), an extension for event processing of the original capabilities of Space Containers, for linking pub/sub systems and mobile peers with short connectivity time. Extending the ideas in [20], this paper reviews existing related work, describes the concept in more detail, and presents first performance evaluation results of the implemented system.

A Space Container allows to store entries in a customizable structured way by means of so called Coordinators. DHTs are used to place such Space Containers in the network in a fault-tolerant and scalable manner. Aspects are components that are triggered whenever a Space Container is accessed. It is a customizable application logic executed either before or after the operation on the Space Container for events processing. An Aspect can be used to check security policies,

to persist, filter or manipulate incoming events, or to alter the content of a Space Container based on the received event. The combination of a pub/sub medium and ASC allow the mobile peer to inset a Space Container in the network which then acts as an intermediate-subscriber for events in the pub/sub medium, and processes the delivered events on behalf of that mobile peer.

The benefits of integrating pub/sub systems and the ASC approach in an ITS scenario are a) less complex application implementation since the processing logic has been moved to the customizable Aspects, b) efficient delivery of events to mobile peers, since relevant information have been extracted while the peer was off-line, thus minimizing the number of messages to be transmitted and avoiding any additional events processing at the peer's site, and c) releases resources of the pub/sub medium since message can be delivered to the intermediate subscriber, the Space Container.

The remainder of this paper is structured as the following: section 2 pictures the use case, section 3 defines the research questions, section 4 summarizes related work, section 5 describes the concept and the architecture of Space Containers and Aspects, while section 6 discusses preliminary evaluation results. Finally section 7 concludes the paper and proposes further work.

2 Scenario

A motivating use-case to identify requirements and to illustrate the benefits of the proposed ASC architecture is an Intelligent Transportation System (ITS) scenario [18]. The scenario consists of a highway with fast moving vehicles communicating with a fixed, geographically distributed infrastructure, as illustrated in figure 1. Along the highway there are so called Road Site Units (RSU) responsible for either passing safety and traffic information to the vehicles or receiving information from the vehicles and pass it to the system. RSUs exchange information via dedicated short range communication protocols (DSRC [21]) and are installed along the road network in 2-3 km distance of each other. They are connected in a meshed wired broadband network in order to assure scalability and increase fault-tolerance, figure 2.

Information exchanged in the system mainly concerns the traffic itself and the messages are published by e.g. the Traffic Control Centre (TCC), radio stations, the police, weather stations, the road maintenance depot, and of course the vehicles themselves. Messages exchanged or events generated depend on the role and may contain information about traffic restrictions and warnings (wrong-way-driver, speed limits, redirections,...), traffic density (the number of cars and their speed within a specific range,...), traffic congestions (location, length, duration, state updates,...), accidents (location, number of cars involved, blocked lanes, state updates,...), road conditions (wet, dry, temperature, number and location of road holes, humidity, hydroplaning warnings,...), current weather conditions (fog) and forecasts, or vehicle related information (acceleration statistics, break hits, sudden use of breaks, average and current speed, passed police control points, car condition, accident alert,...). The published data is geo-located and

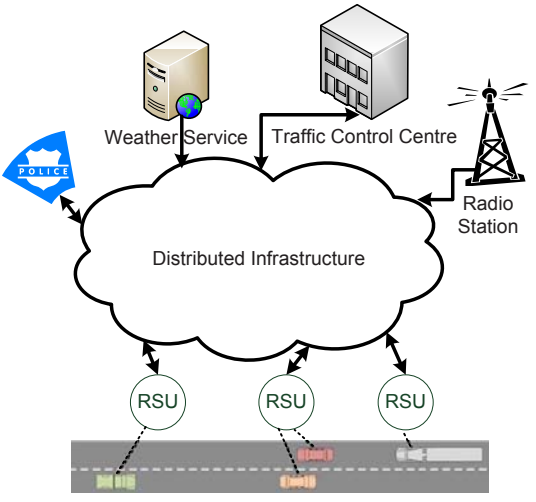

Fig. 1. Publishers and Subscribers in an Intelligent Transportation System

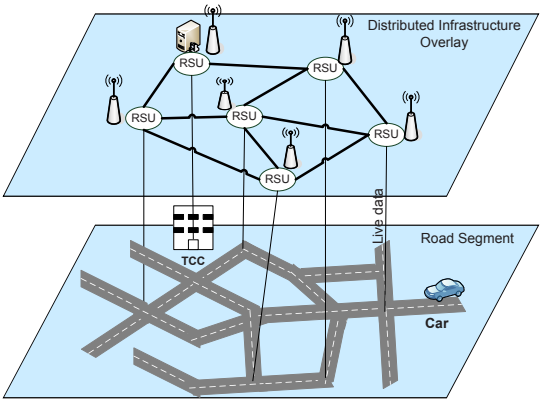

Fig. 2. Mapping of RSUs from the road network onto the meshed network

its relevance in space and time is limited to a certain region, moving direction and period of time. Data belonging to a specific region needs to be queried and updated frequently as vehicles provide new information to the RSU and need the latest data from a RSU situated in the connectivity range.

A subscriber my be a e.g. vehicle driving at high speed or a road worker in field service. Connectivity between the RSU and the passing by vehicles is characterized by a limited bandwidth, communication range, and connectivity window (ca. 300KB/sec for 2-3 sec at 100km/h in case of a single vehicle) allowing the exchange of small and a few messages only [13]. The received events can be used to generate statistics such as about the average speed/lane at coming road segments, the number of vehicles/hour per direction in upcoming road segments, the distribution of vehicles over specific road segments, the average

distance between vehicles, whether groups of vehicles decrease or increase their speed. This kind of information can be used to adapt driving behaviour since drivers are informed about occurrences and actions in upcoming road segments. Road workers in field can use this information to prepare themselves and take precautions ahead in case of increasing traffic density. Road workers may be also interested in statistics like the average temperature and the actual temperature curve of a specific road segment over a specific period of time. Therefore, subscribers are interested in information which a) is represented by the very last event delivered by the pub/sub medium, b) is represented by an aggregated set of events, or c) is a prioritized set of the delivered events. In a special case events can even cancel each other and should not be delivered at all. A vehicle driver does not need to be informed about a wrong-way driver if that driver has already left the road. Additionally, in case of some extraordinary events additional information may be requested from a third person, either to enrich or to verify the event. This would require further network connection.

3 Research Questions

In this paper, we propose the ASC approach of extending the Space Container idea [14] and the concept of distributing Space Containers via Distributed Hash Tables [15], [16], [17], [18] by means of Aspects for efficient pub/sub scenarios in Intelligent Transportation Systems. Based on the limitations of traditional pub/sub communication media with respect to efficient delivery in network environments with mobile peers and on recent projects with industry partners from telematics, we derived the following research questions:

R.1 - The concept of ASC: Investigate the advantages and limitations of the idea of Aspects on top of Space Containers distributed via Distributed Hash Tables, and whether the proposed concept allows reducing the complexity of application implementations. What are the major differences between the aspect-oriented approach and the traditional pub/sub form of durable message delivery?

R.2 - Efficient communication with peers: Investigate to what extent the usage of Aspects in a Space Container helps to decrease communication time within the connectivity time window and as a result increase efficiency of information exchange between mobile peers and the pub/sub medium. Is the proposed concept, combining a DHT lookup and a Space Container access, still fast enough to deliver essential messages aggregated before within the connectivity time window?

For investigating these research issues, we gathered requirements from a set of reasonable industry case studies in the ITS. Then we designed and implemented a framework based on Pastry [19] and the Aspect-oriented Space Container[1] implementation.

[1] To be downloaded at `http://www.mozartspaces.org`

4 Related Work

This section gives an overview about the pub/sub paradigm and its most common (prototype) implementations, and about pub/sub systems in mobile networks with respect to guaranteed delivery. At last, the concept of databases is analysed in order to explain why the concept of Aspect-oriented Space Containers is more appropriate for the scenario than databases.

4.1 Publish/Subscribe Systems

The pub/sub paradigm defines two types of clients: *publishers* generating events, and *subscribers* receiving notifications of events they have previously subscribed for. This type of messaging paradigm allows a strong decoupling of publishers and subscribers in time and space. Furthermore, it enables asynchronous and anonymous communication between publishers and subscribers [1].

There are two types of pub/sub systems: *topic*-based and *content*-based [1]. In a topic-based pub/sub system subscribers can only register their interest on specific topics (mostly predefined) under which publishers are dispatching their generated events/messages. Whereas, in content-based pub/sub systems subscribers are not constrained to specific topics, they can define more precisely in what kind of events/messages they are interested in. On the one hand, this provides more flexibility to the pub/sub system but on the other hand the pub/sub system has to match events/messages to the subscriptions [22]. Hybrid pub/sub systems like Hermes [4], SIENA [23] or REBECA [24], [25] support both types of subscription.

Furthermore, the pub/sub system architecture can be further classified into *client-server* and *peer-to-peer*. In a client-server architecture publishers and subscribers are both clients which are connected to a network of servers. Generally, the servers temporarily store the events/messages generated by publisher-clients and forwarded them to the subscriber-clients. If a subscribed client is not directly connected to a server, where a publisher has dispatched an event, the server has to forward the event to other servers until the event reaches the server capable of delivering the event directly to the subscribed client. Gryphon [26], for example, is a context-based pub/sub system with a client-server architecture, which uses so called brokers as servers. The brokers are responsible to determine which subset of brokers it should send an event/message. In a P2P architecture each node can act as a publisher, subscriber or event-forwarder to another node. SIENA is a mixed form of a client-server and a peer-to-peer architecture, because publishers and subscribers are clients but servers are working together in a peer-to-peer topology.

As mentioned before pub/sub systems enable asynchronous and anonymous messaging between publishers and subscribers. Therefore, such systems have to be reliable and fault-tolerant. Reliable pub/sub systems guarantee that published events/messages are delivered to all its subscribers. Durable (fault-tolerant) pub/sub system are able to cope with unreachable subscribers and servers (due to network failures or crashed clients/servers). Some pub/sub systems like SIENA

offer a best-effort delivery strategy, i.e. the system will periodically retry to deliver the message until the message was delivered successfully, a timeout expired or the maximum retry-count was reached.

The strong decoupling, the asynchronous and anonymous messaging provided by the pub/sub paradigm makes it very attractive to be used in mobile environments [10]. Client applications reside on a host that is moving and therefore accessing the network (composed of so called *event brokers*) from various locations [27]. The event brokers are responsible to guarantee the reliability and durability of the pub/sub system as described before. Furthermore, a protocol must exist which enables the update of a client's subscription as it is moving from one broker to another. During the client's movement undeliverable events have to be stored by the system and delivered as soon as the client reconnects to the system. When the client reconnects at another broker, all stored events have to be forwarded to that broker. The authors of [9] propose a two-phase handover (2PH) protocol, which reduces network traffic and the latency when a client reconnects to the system. One of the first pub/sub system that supports the reconnection of mobile clients is JEDI [28]. Later, existing pub/sub systems like SIENA and REBECA have been extended to support mobile clients [29], [30], [31].

However, our investigations of pub/sub systems have shown that currently most of the available systems are research prototypes which concentrate primarily on scalability and reliability rather on durability in P2P environments. Furthermore, current pub/sub systems for mobile scenarios have the disadvantage that the time needed to update a subscription and to forward all messages to the peer where the re-subscription is made, may take too long. Mobile clients in ITS scenarios are fast moving and have only a few seconds for data transmission (section 2).

In order to adequately address the scaling needs of distributed applications, over the past years there has been research on pub/sub systems making use of the scalability characteristics of Distributed Hash Tables (DHTs) [19]. This has led to several implementations of DHT-based pub/sub systems, like Scribe [32], Meghdoot [33], Willow [34], PastryStrings [35], or [36]. However, the papers aim at using DHT like for routing purposes, extended querying, efficient subscriptions, or the efficient distribution of events. In contrast to those approaches ASC focuses on the distribution of Space Containers as fault-tolerant intermediate subscribers, functioning as a scalable and efficient bridge between mobile peers with very short connectivity time and pub/sub systems. Thus, although ASC has been developed with respect to a mesh network (section 2), it does not prescribe the usage of P2P capable pub/sub systems (section 5.4).

4.2 Databases

The idea of ASC is to combine pub/sub systems with a DHT distributed data storage, the Space Containers, that is capable of aggregating events in order to allow an efficient delivery of events to mobile peers and releasing resources of the pub/sub medium. Therefore, the question is whether databases could have been such a data storage component as well, instead of Space Containers in the described scenario (section 2)? The problem is that established database

products like Oracle or DB2 are heavy-weight components, and as such the peers in the network do not have the capacity to run them. An alternative would be embedded, light-weight databases, like Oracle Berkeley DB Java Edition[2], Apache Derby[3], hsqldb[4], Axion[5], H2[6], or db4o[7]. However, the drawback of databases is that they need a static data model of the entries they have to store, while Space Containers allow the usage of several different Coordinators at the same time, enabling dynamic data models. In case of db4o, being an object-oriented database, accessing an entry is performed via query-by-example, almost like in tuple spaces [37]. However, in [14] it has been shown that Space Containers allow an optimized realization of queries and coordination models.

Finally, databases aim to store and retrieve long living data, while the scenario focuses on short lived data. Additionally, triggers can be seen as aspects as well, but are meant for operations within the database itself, without the power of establishing connections to others peers in the network.

5 Architecture

This section pictures the architecture of an Aspect-oriented Space Container in detail. It describes the interfaces, supported operations, the way of executing operations, and the data-flow between the installed Aspects and the Space Container. Finally, an illustration of the integration between ASC and a pub/sub medium and the use of the architecture to support an Intelligent Application System is described.

5.1 Space Container

A Space Container [14] is a collection of entries accessible via a very few basic methods: read, take, write, shift, and destroy. A Space Container may be bounded to a maximum number of entries, and allows the usage of so called Coordinators. Coordinators enable establishing specific and optimized views on a set of the stored entries. The addressing scheme for a Space Container is an URI of the form `xvsm://mycomputer.mydomain.com:1234/ContainerName`. The Space Container reference is therefore dependent on the IP address of the localhost node that is hosting the Space Container. The protocol type `xvsm` makes the possible communication protocols transparent to the user. Depending on those types, within the platform `xvsm` may be translated to e.g. TCP & Java, specifying that communication takes place via a tcp-connection using java objects. A detailed explanation about Space Containers is given in [14].

[2] `http://www.oracle.com/database/berkeley-db/index.html`
[3] `http://db.apache.org/derby/`
[4] `http://hsqldb.org/`
[5] `http://axion.tigris.org/`
[6] `http://www.h2database.com`
[7] `http://www.db4o.com/`

5.2 Aspects

Space Containers realize some parts of Aspect-oriented Programming (AOP) by registering so called Aspects[8] at different points of a Space Container. Aspects are executed on the peer where the Space Container is located and can be triggered by the various operations on the container. They are triggered by operations either on a specific Space Container or on operations related to the entire set of Space Containers, called Space, rather on the according impact. The join points of AOP are called interception points (IPoints). Interception points, dependent on the Space Container operations are referred to as local IPoints, whereas interception points on Space operations are called global IPoints. Additionally, IPoints can be located before or after the execution of an operation, indicating two categories: pre and post. Therefore, the following local IPoints exist: pre/ post read, pre/ post take, pre/ post destroy, pre/ post write, pre/ post shift, pre/ post aspect appending, pre/ post aspect removing. In addition to the local IPoints the following global IPoints exist: pre/ post transaction creation, pre/ post transaction commit, pre/ post transaction rollback, pre/ post space container creation, pre/ post space container destruction. The main goal of IPoints for aspect appending and aspect removing is to enable exclusive aspects. This means, that an aspect is able to prohibit the addition of new aspects, respectively prohibit itself to be removed.

In addition to the parameters of a Space Container operation a so called Aspect Context can be passed along with every operation allowing the client to communicate with the installed Aspects to make logical decisions or to modify the semantics of the operation completely. Aspect may contain any computational logic, thus can be used to realize security (authorization and authentication), the implementation of a highly customizable notification mechanism, or the additional manipulation of already stored or incoming entries. In case multiple Aspects are installed on the same Space Container, they are executed in the order they were added. Adding and removing Aspects can be performed at any time during runtime.

5.3 Aspect-Oriented Space Container

Figure 3 shows a Space Container with three pre and three post Aspects installed. The accessing operation is executed via the Space Container Interface and passed immediately to the first pre-Aspect. The operation contains the parameters of the operation, like transactions, selectors and timeout, and the Aspect Context. The called Aspect may contain any computational logic that is needed by the user to be executed before the operation. It can either verify or log the current operation, or initiate external operations to other Space Containers or third-party services.

The central part of a Space Container is the implementation of the containers business logic which handles the storage of the entries and coordinates the coordinators. Before an operation can be executed on a Space Container it has to

[8] The complete API JavaDoc can be found at http://www.mozartspaces.org

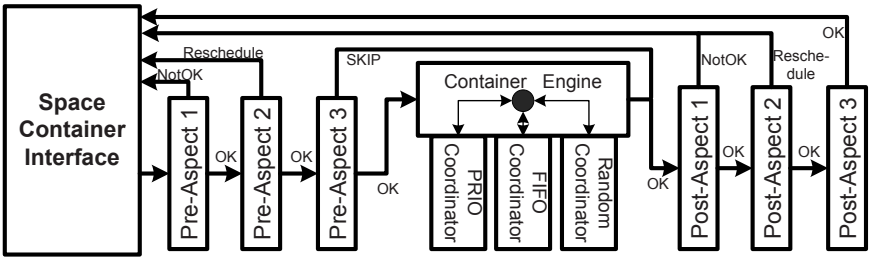

Fig. 3. Execution sequence of Aspects and data- and control-flow in a Space Container with three installed pre- and post-Aspects

pass the installed pre-Aspects. If all Aspects return OK, the container interprets the selectors of the operations and executes the operation [14]. Afterwards, all post-Aspects are executed. Depending on the result of the post-Aspects the result of the operation is either returned to the requesting peer, or the operation is rolled back.

As already mentioned, an Aspect can manipulate the execution of the operation which triggered it. This is realized by the returning values an Aspect can throw. The returned value is analysed and the execution of the operation manipulated accordingly. The following return values are supported:

- **OK:** The execution of the Aspect does not require any changes of the operation, the execution of the operation may proceed normally.
- **NotOK:** The execution of the operation is stopped and the transaction is rolled back. This kind of return value can be used by e.g. a security Aspect denying a operation if the user has not adequate access rights on the Space Container or the Space.
- **SKIP:** The operation is neither performed on the container, nor on the Spaces, nor on any following pre-Aspects. The post-Aspects are executed immediately afterwards. This return value is only supported for pre-Aspects.
- **Reschedule:** The execution of the operation is stopped and will be rescheduled for a later execution. This can be used to delay the execution of an operation until an external event occurs.

5.4 Execution of Aspect-Oriented Space Containers in Publish/Subscribe Scenarios

A reason why we recommend that a Space Container shall be used instead of a database is the fact that the number of different events in an ITS is not known beforehand and as a consequence an appropriate data model is difficult to establish. By means of Coordinators [14], a Space Container is capable of using 'dynamic' data models which can be plugged in whenever needed. A Coordinator allows different views, optimally implemented with respect to accessing requirements, on the entire data in the Space Container at the same time. As

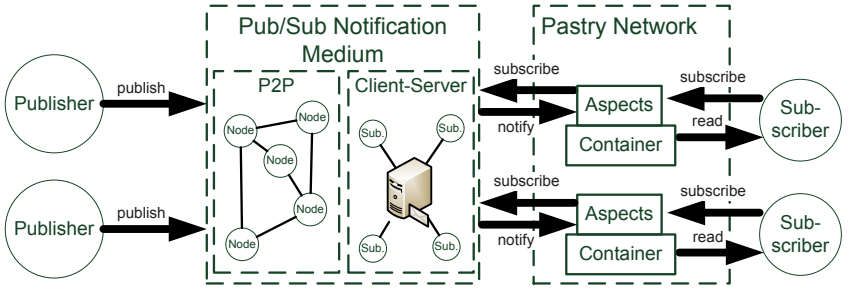

Fig. 4. The operation of Aspect-oriented Space Containers in publish/subscribe scenarios in an Intelligent Transportation System

depicted in figure 4, such a Space Container and its Aspects are deployed by means of a DHT in the RSU network. The principle is described in [17], where we described how to combine Space-Containers with an overlay network based on DHT concepts in order to a) make such Space Containers uniquely addressable in a fault-tolerant and scalable manner, and b) to replicate Space Containers in order to increase fault-tolerance and their availability.

The original subscriber (e.g. a vehicle) places its intention in receiving events from publishers by deploying a Space Container, installing Aspects and publish it in the DHT network. The Aspect registers itself as a subscriber in the pub/sub medium (P2P or client/server style) on behalf and according to the requirements of the original subscriber. From now on the Aspect will, independent of the connectivity mode of the original subscriber, receive events which are then processed by other installed pre-Aspects and the results are then stored in the Space Container. When the original subscriber re-establishes a connection to the network, it uses a read-selector to pick up the results from the Space Container. What kind of selector-type is used is completely up to the original subscriber and depends on the fact how the installed pre-Aspects work with incoming events.

If the Space Container is replicated, Aspects are replicated as well. This means that the original subscriber is subscribed as often as many replicas of that Space Container exists. This is necessary in order to avoid missing events in case one of the replicas, including the subscribed Aspect, is off-line. The way how the replicated Space Containers handle incoming events in order to stay consistent is up to the implementation of the deployed pre-Aspects. Either, the replicas are completely independent of each other and perform every operation as many times as replicas exists, or an incoming event is registered and not used for further processing until the result based on that event has been announced from a designated replica. The latter approach may be more efficient with respect to computational resources but require knowledge about group coordination.

6 Evaluation

In section 3 we defined two research questions: a) can aspects together with space containers reduce the complexity of applications being mostly off-line and

b) can aspects be used for efficient event delivery by decreasing the number of transmitted events and minimizing the communication time required by the applications. Based upon the context of the ITS scenario, introduced in section 2, we implemented a simulation in order to answer the questions.

The problem with complexity is, that it cannot be eliminated but shifted somewhere else and kept abstracted [38]. In case an application could receive all stored events it has subscribed for, it would need the processing logic embedded in the application in order to extract the information the application is really interested in. This processing logic can be outsourced into an Aspect and executed on a Space Container. This implies that application implementation has been reduced in complexity and abstracted by means of the access operations the Space Container concept offers. Additionally, the overall complexity could have been reduced as well. On the one hand each car would have retrieved all events and process them separately. On the other hand, in case of using the ASC approach, processing is done only once at the RSU.

In order to answer the second question we have implemented and evaluated an ITS scenario. Consider an application which is used to monitor the amount of vehicles passing a Road Site Unit in the last 60 seconds and calculate the average speed of these vehicles. In order to realise this functionality, every passing vehicle has to send its current speed to the RSU which stores this information and provides it to vehicles which are interested in it.

We compared the ASC approach with an implementation using durable message queues from the Java Message Service (JMS) [39]. We decided to use JMS because we have not found any downloadable P2P based pub/sub implementations supporting durable subscriptions for our simulation. JMS is an acknowledged API standard developed by Sun Microsystems and implemented by several well-known commercial and non-commercial providers, mainly according to the client-server paradigm. The JMS API provides two messaging models: *queuing* and *pub/sub*. The queueing model is a form of one-way point-to-point communication between a sender and a receiver. The receiver writes messages into a specific queue, wherefrom the receiver consumes the messages in a first-in-first-out manner. The pub/sub model works like the pub/sub paradigm introduced in section 4, but the JMS API specification prescribes that JMS-providers must be implemented reliable and durable.

In order to allow a reasonable comparison of results, the simulation consisted of a single client application, representing the vehicle, and a single peer, representing the RSU. On the one hand the peer was running a JMS server and on the other hand the ASC concept without DHT features, since no distribution is needed. As grounding for our simulation, we used the traffic statistics of the Austrian highway "A23", which is the most frequently used highway in Austria. In 2008, 153100 vehicles used the highway daily[9] in average. Assuming that the amount of cars is consistent over the day, approximately 1.8 vehicles use the highway every second. Additionally, in case RSUs are placed every four

[9] `http://de.wikipedia.org/wiki/ Autobahn_Südosttangente_Wien` last read February, 9th 2009.

kilometres, it means that a vehicle passes the next RSU approximately after 3 minutes, based on the fact that speed is limited to 80 kilometres per hour.

Every time a vehicle drives by a RSU it reports its current speed and (if it is interested in the information) fetches the statistics from the RSU. In the JMS implementation of this scenario we used a durable message queue to store the messages. Every vehicle sends a message containing its current speed to the queue when it is passing by the RSU. Additionally, it reads all the messages which have been published by the other vehicles. Since the JMS standard does not define any way to pre-/post-process messages, the vehicles have to read all messages from the queue, count them to get the amount of cars which passed the RSU and calculate their average speed using the contents of the messages. With the figures presented in the previous paragraph the JMS queue and the physical transmission media has to deal with approximately 50 messages per seconds. The amount of data to be transmitted is limited to 300KB per second in the DSRC protocol. Since the size of the messages is between 5KB to 10KB (depending on the content) it may occur that it is physically impossible to transmit all message from the queue to the vehicle (the connection window of roadside unit is only 2-3 seconds). Since we calculated these figures, assuming the amount of cars is equally spread over the day, the amount might become larger with respect to rush hours and resulting peaks. If it is not possible for a vehicle to retrieve all messages from the message queue the size of the queue grows and any calculations done in the vehicle are not correct.

In the ASC based implementation of this scenario we used an aspect which calculates the average speed. Every vehicle passing the RSU sends its current speed to the RSU. Instead of storing all the messages in the container and providing it for interested parties, an aspect processes the messages and returns the result of the calculation. When a vehicle is interested in the traffic statistics it has to read one message only which contains the aggregated result. Therefore the amount of messages is decreased dramatically.

Figure 5 depicts the increasing amount of data within a message queue in case the messages cannot be retrieved by the vehicles within the connection time windows. The blue line shows the messages when JMS is used. The orange (horizontal) line shows the amount of messages in the ASC based implementation. In case vehicles are in transmission range, they try to fetch as many messages as possible. The only limit is the connection time window. Therefore, every time the blue line falls, messages have been retrieved. However, since - due to low transmission capabilities - not all of the messages could have been transmitted the amount of messages increases continuously. In contrast, the number of messages in the ASC based implementation is always one because only the message which already contains the processed data is stored.

In addition to the simulations described above we implemented several performance tests using ASC with a DHT implementation. Table 1 depicts the time it takes to retrieve a single entry out of 10, 100 and 1000 entries from a Space Container. As shown in the table the time to retrieve the entry is independent of the number of peers (respectively RSUs) in the DHT. This shows that the

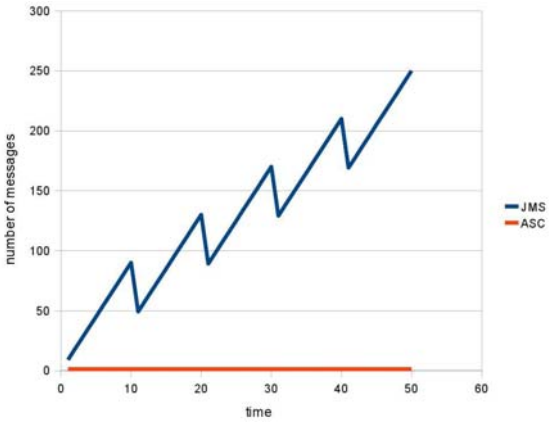

Fig. 5. The development of the size of a message queue in case data cannot be retrieved sufficiently by vehicles within the connection window

Table 1. Durations [ms] for the retrieval of 10, 100 and 1000 entries in a network with 2 to 210 peers [18]

peers	10 entr.	100 entr.	1000 entr.
2	142	208	1378
10	148	258	1002
60	169	256	1115
120	155	265	1184
180	155	231	1086
210	155	231	1086

overhead to lookup the container in the DHT and to execute a query in the Space Container does not have negative influences on the performance of the system. Additionally, the table shows that the time needed to access a Space Container is way lower than the time offered by the connectivity window.

7 Conclusion and Future Work

In this paper we described the concept of ASC, Aspect-oriented Space Containers, distributed via Distributed Hash Tables for bridging mobile peers with small connectivity window and pub/sub systems efficiently in scenarios from the Intelligent Transportation domain. We derived two research questions and answered them based on the evaluation of a scenario from the ITS:

The concept of ASC: The concept of ASC helps moving the complexity of handling events from a peer being mobile and most of the time off-line into so called Aspects of a Space Container. This approach allows the mobile peer to subscribe for events and have the Aspects handle them while the mobile peer is off-line.

Efficiency of the ASC concept: The evaluation showed that the usage of traditional notification media is not sufficient enough to deliver all events to the subscribed peers due to small bandwidth and connectivity time. The operation of Aspects and Space Containers help at preprocessing subscribed events so that the size of the transmitted data and the time needed for transmission is kept minimal. Therefore, the subscriber is capable of receiving all relevant data needed for further decision making.

Future work contains investigations regarding the way how the usage of Space Containers and Aspects change the relation between subscribers and the notification medium. Further questions that we will consider are: Is a durable notification medium still necessary if Space Containers are replicated and distributed via DHTs, since DHT use the same network as the notification medium? Does this have an effect on the semantics of durability? What is the influence on QoS coming from the number of replica set up by the subscriber? Another future work will deal with the question how to move Space Containers along the mesh of RSUs to minimize Space Container access time.

Acknowledgement

This work has been supported by the Complex Systems Design & Engineering Lab, and by the Austrian Government and the City of Vienna within the competence center program COMET.

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. Cabrera, L., Jones, M., Theimer, M.: Herald: achieving a global event notification service. In: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems, 2001, May 2001), pp. 87–92 (2001)
3. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: Scalable application-level anycast for highly dynamic groups. In: Stiller, B., Carle, G., Karsten, M., Reichl, P. (eds.) NGC 2003 and ICQT 2003. LNCS, vol. 2816, pp. 47–57. Springer, Heidelberg (2003)
4. Pietzuch, P.R.: Hermes: A Scalable Event-Based Middleware. PhD thesis, Queens' College University of Cambridge (February 2004)
5. Eugster, P.: Type-based publish/subscribe: Concepts and experiences. ACM Trans. Program. Lang. Syst. 29(1), 6 (2007)
6. Chandramouli, B., Phillips, J.M., Yang, J.: Value-based notification conditions in large-scale publish/subscribe systems. In: VLDB 2007: Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment, pp. 878–889 (2007)
7. Lumezanu, C., Spring, N., Bhattacharjee, B.: Decentralized message ordering for publish/subscribe systems (2006)
8. Pereira, C.M.M., Lobato, D.C., Teixeira, C.A.C., Pimentel, M.G.: Achieving causal and total ordering in publish/subscribe middleware with dsm. In: MW4SOC 2008: Proceedings of the 3rd workshop on Middleware for service oriented computing, pp. 61–66. ACM, New York (2008)

9. Wang, J., Cao, J., Li, J.: Supporting mobile clients in publish/subscribe systems. In: ICDCSW 2005: Proceedings of the First International Workshop on Mobility in Peer-to-Peer Systems (MPPS) (ICDCSW 2005), pp. 792–798. IEEE Computer Society, Washington (2005)
10. Huang, Y., Garcia-Molina, H.: Publish/subscribe in a mobile environment. Wirel. Netw. 10(6), 643–652 (2004)
11. Yoneki, E., Bacon, J.: Dynamic group communication in mobile peer-to-peer environments. In: SAC 2005: Proceedings of the 2005 ACM symposium on Applied computing, pp. 986–992. ACM, New York (2005)
12. Eichler, S.: Performance evaluation of the ieee 802.11p wave communication standard. In: VTC-2007 Fall. 2007 IEEE 66th Vehicular Technology Conference, 2007 (30 2007-October 3 2007), pp. 2199–2203 (2007)
13. Zaera, M.: Wave-based communication in vehicle to infrastructure real-time safety-related traffic telematics. Master's thesis, Telecommunication Engineering, University of Zaragoza (August 2008)
14. Kühn, E., Mordinyi, R., Keszthelyi, L., Schreiber, C.: Introducing the concept of customizable structured spaces for agent coordination in the production automation domain. In: The 8th International Conference on Autonomous Agents and Multiagent Systems (2009)
15. Bessler, S., Tomic, S., Kühn, E., Mordinyi, R., Goiss, H.D.: Sabron: A storage and application based routing overlay network for intelligent transportation systems. In: 3rd International Workshop on Self-Organizing Systems, IWSOS 2008 (2008)
16. Kühn, E., Mordinyi, R., Schreiber, C.: An extensible space-based coordination approach for modeling complex patterns in large systems. In: 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, Special Track on Formal Methods for Analysing and Verifying Very Large Systems (2008)
17. Kühn, E., Mordinyi, R., Goiss, H.D., Bessler, S., Tomic, S.: Integration of shareable containers with distributed hash tables for storage of structured and dynamic data. In: 2nd International Workshop on Adaptive Systems in Heterogeneous Environments - ASHEs 2009, CISIS 2009 (2009)
18. Kühn, E., Mordinyi, R., Goiss, H.D., Bessler, S., Tomic, S.: A p2p network of space containers for efficient management of spatial-temporal data in intelligent transportation scenarios. In: International Symposium on Parallel and Distributed Computing, ISPDC 2009 (2009)
19. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
20. Kühn, E., Mordinyi, R., Keszthelyi, L., Schreiber, C., Bessler, S., Tomic, S.: Introducing aspect-oriented space containers for efficient publish/subscribe scenarios in intelligent transportation systems. In: 8th Working IEEE/IFIP Conference on Software Architecture, WICSA 2009 (2009), `http://tinyurl.com/lx3lmx`
21. Xu, P., Deters, R.: Using event-streams for fault-management in mas. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004 (IAT 2004). Proceedings, September 2004, pp. 433–436 (2004)
22. Liu, Y., Plale, B.: Survey of publish subscribe event systems. Technical report, Computer Science Deptartment, Indiana University (2003)
23. Carzaniga, A.: Architectures for an Event Notification Service Scalable toWide-area Networks. PhD thesis, Politecnico Di Milano (December 1998)
24. Fiege, L.: Visibility in Event-Based Systems. PhD thesis, Technischen Universitt Darmstadt (2004)

25. Zeidler, A.: A Distributed Publish/Subscribe Notification Service for Pervasive Environments. PhD thesis, Technischen Universitt Darmstadt (2004)
26. Bhola, S., Strom, R., Bagchi, S., Zhao, Y., Auerbach, J.: Exactly-once delivery in a content-based publish-subscribe system. In: DSN, pp. 7–16 (2002)
27. Caporuscio, M., Caporuscio, C.M., Carzaniga, A., Carzaniga, A., Wolf, E.L., Wolf, E.L.: Design and evaluation of a support service for mobile, wireless publish/-subscribe applications. IEEE Transactions on Software Engineering 29, 1059–1071 (2003)
28. Cugola, G., Di Nitto, E., Fuggetta, A.: The jedi event-based infrastructure and its application to the development of the opss wfms. IEEE Trans. Softw. Eng. 27(9), 827–850 (2001)
29. Nielsen, J.: Adapting the siena content-based publish-subscribe system to support user mobility. Technical report, Rutgers University - ECE department (2004)
30. Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and evaluation of a wide-area event notification service. ACM Transactions on Computer Systems 19, 332–383 (2001)
31. Muehl, G.: Large-Scale Content-Based Publish/Subscribe Systems. PhD thesis, TU Darmstadt (2002)
32. Castro, M., Druschel, P., Kermarrec, A.M., Rowstron, A.: Scribe: a large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications 20(8), 1489–1499 (2002)
33. Gupta, A., Sahin, O.D., Agrawal, D., Abbadi, A.E.: Meghdoot: content-based publish/subscribe over p2p networks. In: Jacobsen, H.-A. (ed.) Middleware 2004. LNCS, vol. 3231, pp. 254–273. Springer, Heidelberg (2004)
34. van Renesse, R., Bozdog, A.: Willow: Dht, aggregation, and publish/subscribe in one protocol (2005)
35. Aekaterinidis, I., Triantafillou, P.: Pastrystrings: A comprehensive content-based publish/subscribe dht network. In: ICDCS 2006: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems, p. 23. IEEE Computer Society, Washington (2006)
36. Ahull, J.P., Lpez, P.G., Skarmeta, A.F.G.: Caps: Content-based publish/subscribe services for peer-to-peer systems. In: 2nd Int. Conf. on Distributed Event-Based Systems, DEBS 2008 (2008)
37. Gelernter, D.: Generative communication in linda. ACM Trans. Program. Lang. Syst. 7(1), 80–112 (1985)
38. Brooks Jr., F.P.: The mythical man-month (anniversary ed.). Addison-Wesley Longman Publishing Co., Inc., Boston (1995)
39. Hapner, M., Burridge, R., Sharma, R., Fialli, J., Stout, K.: Java message service. Technical report, Sun Microsystems, Version 1.1 (April 12, 2002)