A P2P Network of Space Containers for Efficient Management of Spatial-Temporal Data in Intelligent Transportation Scenarios

Eva Kühn, Richard Mordinyi, Hannu-Daniel Goiss
Complex Systems Design and Engineering Lab
Vienna University of Technology
Favoritenstr. 9-11, 1040 Vienna, Austria
{*eva, rm, hdg*}*@complang.tuwien.ac.at*

Sandford Bessler, Slobodanka Tomic
Telecommunications Research Centre Vienna
Donau-City 1, 1210 Vienna, Austria
{*bessler, tomic*}*@ftw.at*

Abstract

The effectiveness of Intelligent Transportation Systems (ITS) depends on their ability to collect contextual data from various sources and appropriately generate and transport comprehensible, reliable and timely content to users. In such applications, the exchanged content is structured in space and time. Peer-to-peer (P2P) networks are the natural choice these applications due to their fault-tolerance, self-organization and scalability properties. However, a closer analysis of the available Distributed Hash Tables (DHT) protocols shows that the structure of the data gets lost and its short liveness leads to high signalling traffic.

In this work we propose a novel overlay network of so called Space Containers for storing, accessing, manipulating and structuring dynamic geo-located content. The benefits of combining Space Containers and DHT are: clean application programming logic and efficient content retrieval while preserving the properties of DHTs.

We describe the system architecture applied to a transportation scenario and show preliminary evaluation results.

1. Introduction

Intelligent Transportation Systems (ITS) [1] are decision support systems that offer assistance in terms of instructions and recommendations to drivers, describing road and traffic conditions, and that operate in a dynamically changing environment. Therefore, the effectiveness of such applications depends on the ability to collect contextual data from many different sources such as sensors, cameras or personnel, and to appropriately generate and transport comprehensible, reliable and timely content to users. Due to the high vehicle mobility, ITS applications require novel opportunistic routing, node-by-node reliable transport and disruption-tolerant behavior. In order to meet these requirements intermediate nodes must be able to cache and replicate the communication messages.

The trend for "storage in the net" has been efficiently addressed by P2P storage networks such as PAST [2], OceanStore [3], or the cooperative file system CFS [4]. However, when it comes up to efficiently storing, querying, or acting-upon structured content, such as contextual data including complex temporal as well as geographical location information, the former storage network designs (supporting mainly files) becomes unsuitable, since the information structure gets lost.

In [5] we proposed a novel architecture which integrates the adaptiveness of the P2P approach, particularly Distributed Hash Tables (DHT), with the query expressiveness and coordination support of the Space-based Computing paradigm [6]. The so called SABRON [7] (storage and application based routing overlay network) architecture enables applications that operate on distributed storage of structured data and that require application-based routing and distributed coordination. SABRON combines the so called Space-Containers [8] for storage, access, and manipulation of complex, dynamic data objects, while an overlay network based on DHT concepts makes such Space Containers uniquely addressable in a fault-tolerant and scalable manner. The benefits of SABRON are: a) low message traffic on the DHT level, b) clean application programming interfaces that hide the distributed character of space containers, and c) efficient data storage and retrieval due to customizable coordinators and query expressiveness of Spaces-Containers. Extending the ideas in [5], this paper reviews existing related work and presents first performance evaluation results of the implemented system.

The remainder of this paper is structured as follows: section 2 summarizes related work, section 3 defines the research questions, and section 4 presents an application scenario describing the requirements and motivating our approach. Section 5 describes the concept and the architecture, whereas section 6 discusses the evaluation results. Finally section 7 concludes the paper and proposes further work.

978-0-7695-3680-4/09 $25.00 © 2009 IEEE
DOI 10.1109/ISPDC.2009.27

2. Related Work

This section summarizes related work on Space-based Computing and DHT concepts with focus on distribution and retrieval of dynamic data in P2P networks. It shows the benefits and strengths of each technology, vis-a-vis of the requirements of vehicular applications.

2.1. Distributed Hash Table for Lookup

Structured P2P networks, like Chord [9], Pastry [2], or CAN [10], use Distributed Hash Tables and have been extensively studied and successfully deployed to create scalable and fault-tolerant applications.

A few P2P techniques such as the intentional naming system INS/Twine [11] or P-Grid [12] can maintain a structure with a hierarchical DNS-like addressing. However, the costs for such solutions are high, since keeping a P2P entry for each data item (or message) causes a large traffic overhead, especially when the data elements are mutable and short lived. The basic functions a DHT provides, are to store a data object and to retrieve it efficiently from any peer in the network. For this purpose, a very large identifier space (e.g. all binary combinations of 128 bits) is defined onto which data objects and node identifiers are mapped using a hash function. The design options to create the *key* for a data object vary from hashing the (string) name of the object, its location (URI), or the content itself. The storage/publishing function is not affected by the selected alternative *publish(key, value)*.

With most DHTs it is only possible to lookup exact values, but it is not possible to include wildcards or any other query expression for retrieving values. DHTs take care of efficient distribution of the data in the key space for load sharing purposes, and of replication for fault-tolerance reasons. Therefore, a DHT offers basic self organization functions, such as adapting the topology when a node arrives or leaves, and a better reliability since it has a mechanism in place to replicate DHT entries. The lookup effort is for most DHTs O(log N) where N is the number of nodes.

Among the various DHT algorithms, the Pastry protocol [2] with its implementation FreePastry[1] was used to evaluate our approach and deserves a more detailed discussion. The identifiers are selected from a 128-bit ID circular space. The routing of messages between the nodes is based on maximum prefix matching. For this purpose the routing table of a node has several rows, so that the n-th row lists those nodes that have a matching prefix of n with the current node ID. The leaf set is also a part of the routing table and contains nodes whose node ID are numerically close to the current node ID. DHT entries are replicated by Pastry on a subset of the leaf set. The Pastry replication mechanism is responsible that the parameter *r*, the number of life replicas, remains invariant under node churn conditions. During a lookup operation, the routing protocol will search the node with the closest ID to the key K, however a neighbor node holding *replica(K)* might be found earlier on the routing path. Therefore, the *lookup(k)* operation might not deliver deterministically the same replica.

1. http://freepastry.org/FreePastry/

2.2. Space-based Computing for Storage and Retrieval

The Linda coordination model [13], developed in the mid-1980's by David Gelernter at Yale University, is the originator of the "space based system". He introduced a coordination language called Linda which operates on an abstract computation environment called tuple space. In the tuple space approach, processes communicate with other entities in the environment by writing tuples (ordered sequences of data) into and reading tuples from the tuple space via a handful operations (*out*, *in*, *rd*, *eval*). Sharing of data via spaces [14] is not a novel paradigm. It comes from parallel processing and was later considered for distributed environments. Due to its high-level abstraction of communication by simply reading and writing data from/into a shared space this paradigm fits to growing dynamics and collaboration in the network [15].

There are a lot of implementations (like JavaSpaces [16], TSpaces [17], LIME [18], MARS [19], or TuCSon [20]), that follow the concept of the tuple space with associative search for the stored tuples. The Linda model requires the specification of a tuple as an argument for both query operations. In such a case, the tuple is called template that allows the usage of a wildcard as the field's value. A wildcard declares only the type of the seeked field, but not its value. Although both MARS and TuCSoN enable the modification of the operations' semantics by adding so called *reactions*, they can not influence the way how tuples are queried. JavaSpaces adds subtype matching to the exact tuple matching mechanism to query objects from the space. The drawback of exact tuple matching is that all collaborating processes must be aware of the tuple's signature they use for information exchange. Hence, there are several tuple space implementations that offer additional queries mechanisms, such as TSpaces [17], XMLSpaces.Net [21] and eLinda [22]. TSpaces offers the possibility to query tuples by named fields or by specifying only the field's index and a value or wildcard. Furthermore TSpaces allows the definition of custom queries by introducing the concept of factories and handlers. Both TSpaces and XMLSpaces.Net support the use of XML-documents in tuple fields and therefore enable the use of several XML query languages such as XQL or XPath. eLinda [22] enables the usage of more flexible queries, via its Programmable Matching Engine (PME), such as maximum or range queries. Beside these queries the PME

also provides *aggregated operations* that allow the summary or aggregation of information from a number of tuples, returning the result as a single tuple. The PME allows, like TSpaces with its concept of custom factories and handlers, the simple definition of custom matchers [22]. However, it cannot be guaranteed which tuple is returned by a query. It may happen that due to the non-deterministic semantics of the Linda operations a tuple is never returned although it would match a query.

The SABRON approach uses a Space-based architecture called XVSM (extensible virtual shared memory [23]) and its implementation called MozartSpaces[2]. A part of MozartSpaces is an abstraction called Space Containers [8], [24]. A Space Container allows the storage of entries in a customizable structured and ordered way. Entries are data structures of any type, not being restricted to tuples only. The ordered, structured form of the space is achieved by specific customizable Coordinators, an inherent component of a Space Container. Such Coordinators are capable of distinguishing explicitly between data needed for coordination purposes and the payload itself. Those coordination data is used by the Coordinators to store the entries in an efficient way. Like in the Linda model, Space Containers can be accessed by the operations *read*, *take*, and *write*. Each Space Container may contain one or more Coordinator which define the exact semantics of each operation. The Coordinators can be classified into *implicit order*, *direct access* and *content matching*. Implicit order refers to e.g. FIFO and LIFO ordering of stored entries. Direct access allows selection of entries via tags directly, using relational operators [24] or range operators. Content matching allows to define user defined match makers [25], e.g. for Linda, RDF, or XML querying facilities.

A data space in MozartSpaces is a collection of Space Containers which can be addressed via URLs in the internet. A lookup mechanism has to resolve a published container name to its URL. Depending on the application domain, the use of DHT techniques as described in section 2.1 is a candidate to consider. Additionally, a Space Container is extensible in the sense that its behavior can be extended by means of so called Aspects [26], comparable to aspects in object-oriented programming languages [27]. Aspects represent application programs that are executed on certain operations carried out on a Space Container. Interception points (e.g. pre- or post-read) for Aspects define whether the logic is carried out before or after every operation.

A MozartSpace was originally conceived as a coordination framework, but can be used as a storage component as well, similar to a distributed database. The customizable Coordinators leading to extensive query expressiveness, the Aspects that react on events, are all benefits that distributed databases however do not have at this level. Databases aim to store and retrieve long living 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. Furthermore, distributed databases are capable of replicating data in a very efficient way, but lack integrating the semantics of data to be replicated into that process. Consequently, they cannot support different replication and consistency strategies based on the semantics of the stored data at the same time (section 5.2). Finally, databases support static data models while Space Containers allow the usage of several different Coordinators at the same time, enabling dynamic data models.

3. Research Questions

The SABRON approach combines the DHT addressing and networking concepts with Space Containers for efficient storage and retrieval of dynamic and structured data. Due to the limitations of traditional DHT approaches with respect to structured and dynamic data and to performance requirements in vehicular transportation environments, we derive the following research questions:

R.1 - Functional and programming aspects: Investigate a) the advantages and limitations of an integration of Space Containers and a DHT network for storing, manipulating, and retrieving structured and dynamic data; b) system self-organization and resilience properties and c) programming easiness and elegance combined with added support of more complex data queries. What are the major differences between the integrated approach and a ”plain” DHT solution?

R.2 - System Performance: Investigate to what extent the integration of Space Containers with DHTs a) reduces the time needed to collect all data belonging to the same domain, b) supports queries for precise retrieval of data. Where are the strengths and weaknesses of each concept? For investigating these research issues, we gathered requirements from a set of reasonable industry case studies in the telematics domain. Then, we designed and implemented a framework based on Pastry [2] and our Space Container[3] implementation.

4. Application Scenario

A motivating use-case that we use to identify requirements and to illustrate the benefits of the proposed architecture is an Intelligent Transportation System (ITS) scenario. In this scenario fast moving vehicles communicate with a fixed, but geographically distributed infrastructure, as illustrated in figure 1.

For this purpose the peer nodes are called road side units (RSUs), and are installed along the road network in 1-2 km distance of each other. RSUs are connected in a meshed

2. to be downloaded at http://www.mozartspaces.org

3. to be downloaded at http://www.mozartspaces.org

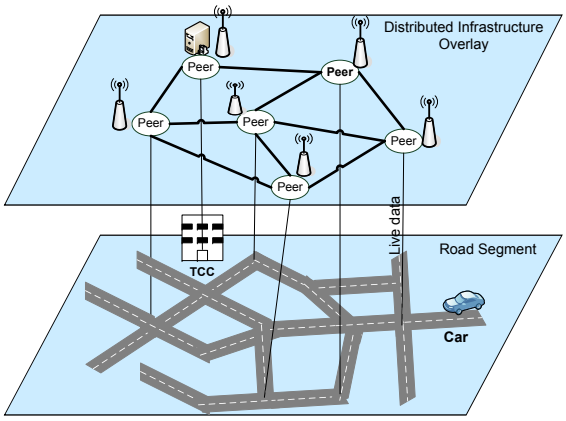

Figure 1. Intelligent Transportation Scenario

wired broadband network in order to assure scalability and increase fault-tolerance. On the one hand RSUs exchange safety and traffic information via dedicated short range communication protocols (DSRC [28]) with vehicles passing by. Such geo-located messages include safety-critical warnings like wrong-way-driver, speed limits, traffic jams, incidents, road works etc. One service offered by the road provider is to distribute these messages to *the relevant* RSUs. Depending on the type and severity of the message, the relevant RSU is one that is situated within a certain distance ahead (or upstream) of the event (e.g. an accident).

On the other hand the vehicles themselves act as providers of geo-temporal context data by reporting on safety events measured by sensors along their paths, e.g. sudden use of breaks, humidity, temperature of the road, etc. In both cases the scenario shows that the data is geo-located and 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.

5. Architecture

As mentioned before, the proposed system should keep the structure of information entities while preserving the characteristics of DHTs. Thus, grouping can be achieved by storing the information entities in a Space Container. This means that the addressing granularity in the overlay is therefore the Space Container. Each Space Container may represent a geographical region, a certain service, or the messages a RSU has to broadcast to all vehicles passing by. 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.

In case the IP address changes, a node fails or the containers are redistributed on other peers for better load-sharing, Space Containers cannot be addressed, unless a redirection at the DHT level made is feasible. The indirection layer added by means of the DHT makes possible to hide IP address changes. Once the current IP address is obtained via DHT, the Space Container is addressed directly via IP routing and allows clients to lookup Space Containers by *container name* only.

5.1. Managing Space Container replication through Pastry

Space Containers need to be replicated in order to increase reliability, availability and to minimize data access time. In case a RSU is temporarily off-line, its replicated Space Containers can still be updated. This allows the seamless continuity of operations without data loss when the failed RSU is on-line again. The Pastry replication algorithm is based on an invariant number of replicas of an object addressed by a key K, on r nodes that are alive and nearest to key K. Similarly to the PAST storage system built on top of Pastry [29], replicas have to be managed in order to fulfill the invariant rule mentioned above. In the following discussion we distinguish between replicas of DHT entries (DHT replicas) and replicas of the Space Containers (container replicas).

An key-value pair entry in DHT looks like: *key* = *H(containerName)*, *value* = *containerReferenceURI*. The Pastry routing protocol makes sure that an alive replica is found. However, finding the DHT entry is not enough, its value has to point to the correct container replica. We describe how to correctly maintain the replicas. Let's consider first the replication of DHT entries: Pastry can always return the *replicaSet* for a certain key, that is a set of nodes that can be used to store replicas of that key. On each of these nodes a replica of the "stripped" container reference is stored under the key, that is the *ContainerReferenceURI* without the first part that defines the domain name. When reading this Value, the local node (the destination node of the lookup) completes the URI with the IP of the localhost, before returning it to the requesting client. This solution has two consequences:

- The DHT entries are the same for all replicas, thus they do not have to be manipulated or rewritten in failure case - a substantial advantage.
- Replicated DHT entries and the corresponding containers have to be on the same node.

In case a node fails, the lost DHT entries are replicated in order to keep the number of copies for each entry equal to *r*. Specifically, in case the number of replicas is less than r, a DHT entry replica is automatically created by Pastry on those nodes from the set *replicaSet(K)* that have no replica. In addition to this DHT mechanism, the container corresponding to the newly created replica has to be copied as well on that same node.

5.2. Handling container replicas

Space Containers are passive components in which the entries change following operations such as write, take, or destroy. Therefore, the container replicas have to be updated, although the DHT pointers remain unchanged. It has to be noted that the mechanisms that replicate DHT entries and Space Containers are decoupled. Whereas the former depends on the implementation of the DHT protocol, the latter depends on the implementation of the Aspects placed on a Space Container. When the replication occurs at DHT level, it triggers the replication operation at Space Container level.

The design of the interactions related to a WRITE operation has to consider the replication mechanisms specific for the DHT implementation - in our case - Pastry. In the normal case the DHT operation *lookup(K)* will access the root node, but if the path to destination hits first a replica node, it is this node that is returned. Therefore, in this system we do not have a replication with a single master. We propose to use a delegation pattern as shown in figure 2. The first interaction *lookup(K)* will return the full address of a container replica (or the root). Within a Space Container transaction, the requester peer proceeds with a direct Space Container *write()* operation, followed by an *Aspect* that calls *replicaSet()* in order to obtain the other replica nodes, and repeats the *write* operation at the respective Space Containers. The transaction to the first accessed Space Container ends with the result. Figure 2 shows the interactions.

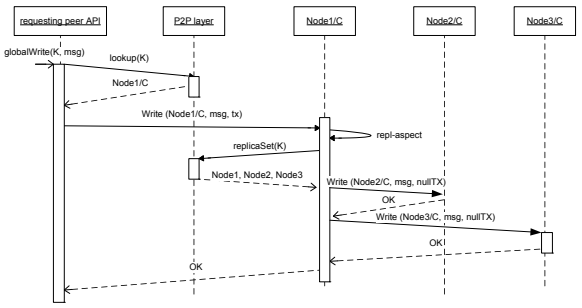

Figure 2. Handling container replicas

5.3. Handling joining and leaving nodes

If a node holding a key K is non responsive for a certain period of time, e.g 10 seconds, then Pastry triggers an adjustment of the leaf set routing entries in all affected nodes. Each node removes the failed node from its leaf set and includes the live node with the closest *nodeId*. The responsibility for the key K moves from the failed node to another near node that however has first to acquire a DHT replica of K. As we have seen in the previous section, a new DHT replica of K triggers the creation of new container replica on the same node.

If a new node joins the system, this node is included in the leaf set of L neighbors, and other nodes are removed from those leaf sets. If the key K points to one of these nodes, there remain only *r-1* replicas, and the new node has to require a replica for K. Therefore, the overall effect of a "join" is to move the container from the node dropped from the leaf set, to the new node.

6. Evaluation

In order to answer the questions related to system performance in section 3, we have implemented the system and performed the following preliminary tests:

- test case A: retrieve all safety messages that belong to a specific road segment (being collected in one Space Container).
- test case B: retrieve a specific message of a specific road segment.

In Figure 3 the diagrams show the sequence of operations for case A: on the left side a plain DHT is used to store each message under a different key. In the plain-DHT setup (left), the vehicle client retrieves first the keys stored under a well-known key (KW) corresponding to the road segment name. Then, the individual keys are used to retrieve the messages. The following operations have to be executed in the presented order:

```
//Op. 1.: lookup value of well-known key
values = lookup(KW);
//Op. 2-9: get values of received keys
while ((key = nextKey(values)) != FALSE) {
  info = lookup(key);
}
```

On the right side of Figure 3 the DHT entry points to a Space Container that contains a number of data messages. This implementation requires the execution of only two operations: the first operation retrieves the Space Container reference URI (CREF), by means of a well-known key (CNAME) via the DHT. The second operation retrieves the entries in the container found at the node given by that URI. As it can be seen, the query complexity has been shifted from the DHT to the Space Container level.

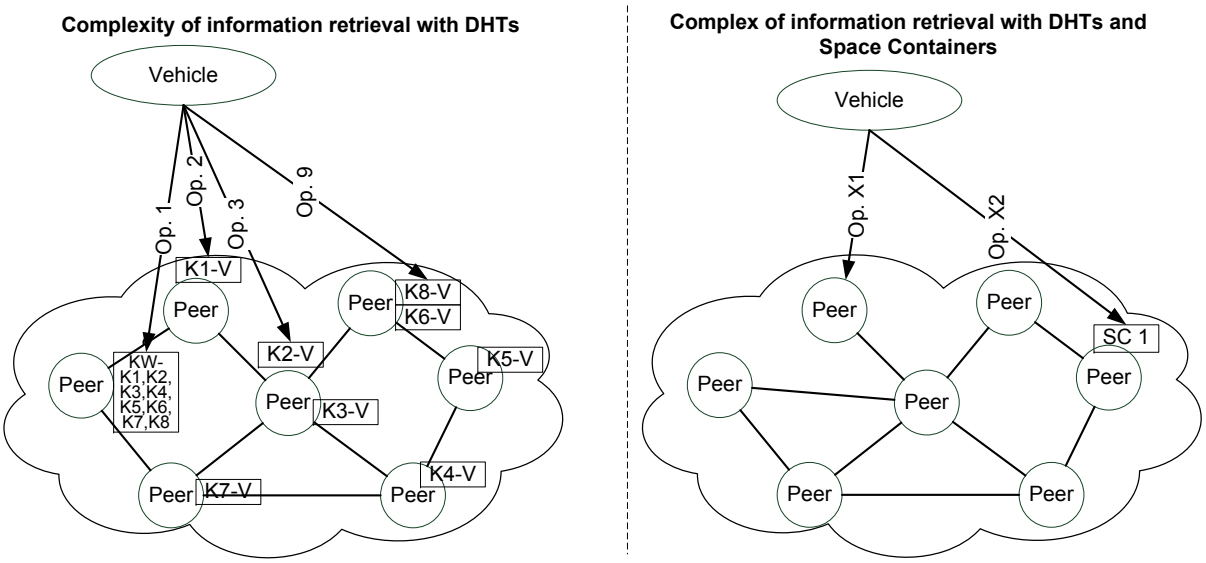

Figure 3. Comparing the complexity of retrieving information with DHTs only with the concept of integrating DHTs and Space Containers

```
// get Space Container reference
Op. X1.: CREF = lookup(CNAME);
// get all entries
Op. X2.: read(CREF, new Selector(All))
```

In order to quantify the efficiency of the proposed approach, we measured the time needed to run the test cases A and B, using the WINZIG Grid [30] at the Vienna University of Technology. The Grid consists of 300 standard desktops clustered in groups of 30 machines. The groups are interconnected with high-speed switches organized hierarchically. The task in test case A is to retrieve all 10, 100, or 1000 messages that belong to a specific road segment based on a network with between 10 and 210 peers and measure the time needed to do so. The setup for test case B is the same, but in this case a single message matching a specific query is returned.

configuration	peers	10 entr.	100 entr.	1000 entr.
plain DHT	2	0.3	4.4	61.3
plain DHT	10	283	2485	24529
plain DHT	60	263	2455	24126
plain DHT	120	277	2586	24153
plain DHT	180	264	2491	24252
plain DHT	210	263	2613	24098
DHT+XVSM	2	142	208	1378
DHT+XVSM	10	148	258	1002
DHT+XVSM	60	169	256	1115
DHT+XVSM	120	155	265	1184
DHT+XVSM	180	155	231	1086
DHT+XVSM	210	155	231	1086

Table 1. Durations [ms] for the retrieval of 10, 100 and 1000 entries in test case A

Table 1 shows the result for case A. The measured overall lookup-time consists of a lookup for the list of keys (stored under a well-known key), and additional 10, 100, and 1000 lookups for each of these keys. In total, a number of 11, 101, and 1001 lookups have to be performed, where a single lookup takes about 25 ms. As we can see, the increase in the path length with the number N of peers (which is proportional to O(log N)) does contribute little to the lookup time in the plain DHT case. It is the request and response processing that accounts for most of the 25 ms needed by each lookup.

In this scenario the caching capabilities of the DHT implementation were disabled (to avoid returning invalid values due to the high number of information updates).

For the space container architecture (DHT+XVSM) the measured times are much lower, because there are only two operations to perform: a first lookup retrieves the Space Container reference, and the second direct request fetches 10, 100 or 1000 entries at once.

Table 2 shows the retrieval time of one entry in the XSVM system by directly addressing the container (i.e. test case B, without the DHT lookup time). Different pattern matching mechanisms have been compared, starting with the classical Linda tuple case, an improved query and prioritized FIFO buffer. The low numbers show that the improved algorithms account for a small part of the total time in the XSVM experiments of Table 1.

Entries	Linda	Improved-Linda	PRIO-FIFO
10000	5.24ms	0.41ms	0.20ms
20000	15.15ms	0.50ms	0.20ms
30000	47.93ms	0.57ms	0.21ms
40000	58.66ms	0.63ms	0.20ms
50000	70.10ms	0.66ms	0.21ms

Table 2. Time to retrieve a single entry using different Space Container access modes (coordinators)

7. Conclusion and Future Work

In this paper we describe SABRON, a concept of integrating DHTs and Space Containers, which allows preserving the spatial-temporal structure of short lived data messages, and benefits of the characteristics of DHTs like fault-tolerance, self-organization and scalability. In the process of design and implementation of the system, and later in preliminary experiments we answered two research questions referring to programming ease, processing complexity and the potential of the architecture to build powerful application based message routing and storage functionality in high mobility scenarios.

Concept of SABRON: DHTs have a few known drawbacks: similarity and locality of data. Without specifying exactly the searched name or range, a searche cannot be performed without special handling in the application. Using Space Containers we can preserve the spatial proximity of the stored events, an important requirement for the management of geo-temporal data. Furthermore, manipulation of data in a container does not create DHT signalling traffic since the container links do not change. In a future version of the system the Space Container replication strategy could be adapted to the needs of the application and be independent of the DHT replication mechanism.

Efficiency of SABRON. The evaluation shows that the integration of DHTs and Space Containers improves the efficiency of information retrieval. This is due to the fact, that the SABRON approach requires only two operations to be executed. The first one, to lookup the Space Container, and the second one to execute the query. Further work will provide performance measurements on an extended prototype that will show the behavior of SABRON on a internet-wide P2P platform like PlanetLab in presence of node churn.

Acknowledgement

This work has been supported by the Austrian Government and by the City of Vienna within the competence center program COMET.

References

[1] J. Sussman, *Perspectives on Intelligent Transportation Systems (ITS)*. Springer, New York, NY, 2005.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," *Lecture Notes in Computer Science*, vol. 2218, pp. 329–??, 2001. [Online]. Available: citeseer.ist.psu.edu/rowstron01pastry.html

[3] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and B. Zhao, "Oceanstore: an architecture for global-scale persistent storage," *SIGOPS Oper. Syst. Rev.*, vol. 34, no. 5, pp. 190–201, 2000.

[4] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with cfs," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 202–215, 2001.

[5] E. Kühn, R. Mordinyi, H.-D. Goiss, S. Bessler, and S. Tomic, "Using tuple-spaces to build a storage p2p system for structured and dynamic data," *2nd International Workshop on Adaptive Systems in Heterogeneous Environments - ASHEs'09, CISIS 2009*, 2009.

[6] P. Ciancarini, "Distributed programming with logic tuple spaces," *New Gen. Comput.*, vol. 12, no. 3, pp. 251–284, 1994.

[7] S. Bessler, S. Tomic, E. Kühn, R. Mordinyi, and H.-D. Goiss, "Sabron: A storage and application based routing overlay network for intelligent transportation systems," *3rd International Workshop on Self-Organizing Systems, IWSOS 2008*, 2008.

[8] E. Kühn, R. Mordinyi, L. Keszthelyi, and C. Schreiber, "Introducing the concept of customizable structured spaces for agent coordination in the production automation domain," *Accepted for the 8th International Conference on Autonomous Agents and Multiagent Systems - AAMAS 2009 (TechRep at http://www.complang.tuwien.ac.at/richard/techrep/aamas09.pdf)*, 2009.

[9] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, p. 1732, 2003.

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker, "A scalable content-addressable network," in *SIGCOMM'01: Proceedings of the 2001 conference on Applications,technologies,architectures,and protocols for computer communications*. New York,NY,USA: ACM Press, 2001, p. 161172.

[11] M. Balazinska, H. Balakrishnan, and D. Karger, "Ins/twine: A scalable peer-to-peer architecture for intentional resource discovery," pp. 195–210, 2002.

[12] K. Aberer, "P-grid: A self-organizing access structure for p2p information systems," in *CooplS '01: Proceedings of the 9th International Conference on Cooperative Information Systems*. London, UK: Springer-Verlag, 2001, pp. 179–194.

[13] D. Gelernter, "Generative communication in linda," *ACM Trans. Program. Lang. Syst.*, vol. 7, no. 1, pp. 80–112, 1985.

[14] N. Carriero and D. Gelernter, "Linda in context," *Commun. ACM*, vol. 32, no. 4, pp. 444–458, 1989.

[15] Z. Li and M. Parashar, "Comet: A scalable coordination space for decentralized distributed environments," in *HOT-P2P '05: Proceedings of the Second International Workshop on Hot Topics in Peer-to-Peer Systems*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 104–112.

[16] E. Freeman, K. Arnold, and S. Hupfer, *JavaSpaces Principles, Patterns, and Practice*. Essex, UK, UK: Addison-Wesley Longman Ltd., 1999.

[17] P. Wyckoff, S. W. McLaughry, T. J. Lehman, and D. A. Ford, "T spaces," *IBM Systems Journal*, vol. 37, no. 3, pp. 454–474, 1998.

[18] A. L. Murphy, G. P. Picco, and G.-C. Roman, "Lime: A coordination model and middleware supporting mobility of hosts and agents," *ACM Trans. Softw. Eng. Methodol.*, vol. 15, no. 3, pp. 279–328, 2006.

[19] G. Cabri, L. Leonardi, and F. Zambonelli, "Mars: a programmable coordination architecture for mobile agents," *Internet Computing, IEEE*, vol. 4, no. 4, pp. 26–35, Jul/Aug 2000.

[20] M. Cremonini, A. Omicini, and F. Zambonelli, "Coordination and access control in open distributed agent systems: The tucson approach," pp. 369–390, 2000. [Online]. Available: http://dx.doi.org/10.1007/3-540-45263-X_7

[21] R. Tolksdorf, F. Liebsch, and D. M. Nguyen, "Xmlspaces.net: An extensible tuplespace as xml middleware," in *In Report B 03-08, Free University Berlin, ftp://ftp.inf.fu-berlin.de/pub/reports/tr-b-0308.pdf, 2003. Open Research Questions in SOA 5-25 and Loose Coupling in Service Oriented Architectures*, 2004.

[22] G. Wells, A. Chalmers, and P. Clayton, "Extending the matching facilities of linda," in *COORDINATION '02: Proceedings of the 5th International Conference on Coordination Models and Languages*. London, UK: Springer-Verlag, 2002, pp. 417–432. [Online]. Available: http://www.springerlink.com/content/hqp9ubltnqnu2b00

[23] E. Kühn, J. Riemer, R. Mordinyi, and L. Lechner, "Integration of xvsm spaces with the web to meet the challenging interaction demands in pervasive scenarios," *Ubiquitous Computing And Communication Journal (UbiCC), special issue on "Coordination in Pervasive Environments"*, vol. 3, 2008.

[24] E. Kühn, R. Mordinyi, and C. Schreiber, "An extensible space-based coordination approach for modeling complex patterns in large systems," *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.

[25] G. P. Picco, D. Balzarotti, and P. Costa, "Lights: a lightweight, customizable tuple space supporting context-aware applications," in *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2005, pp. 413–419.

[26] E. Kühn and F. Schmied, "Xl-aof: lightweight aspects for space-based computing," in *AOMD '05: Proceedings of the 1st workshop on Aspect oriented middleware development*. New York, NY, USA: ACM, 2005.

[27] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin, "Aspect-oriented programming," in *Proceedings European Conference on Object-Oriented Programming*, M. Akşit and S. Matsuoka, Eds. Berlin, Heidelberg, and New York: Springer-Verlag, 1997, vol. 1241, pp. 220–242. [Online]. Available: citeseer.ist.psu.edu/article/kiczales97aspectoriented.html

[28] Q. Xu, T. Mak, J. Ko, and R. Sengupta, "Vehicle-to-vehicle safety messaging in dsrc," in *VANET '04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*. New York, NY, USA: ACM, 2004, pp. 19–28.

[29] A. I. T. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "Scribe: The design of a large-scale event notification infrastructure," in *Networked Group Communication*, 2001, pp. 30–43. [Online]. Available: citeseer.ist.psu.edu/rowstron01scribe.html

[30] P. Kolmann, "University campus grid computing," Master's thesis, Vienna University of Technology, 2005.