SORPA: A Self-Organizing Data Replication Algorithm for Smart Products

Markus Miche
SAP Research Switzerland
Kreuzplatz 20
CH-8008 Zürich, Switzerland
markus.miche@sap.com

Ivan Delchev
SAP Research Switzerland
Kreuzplatz 20
CH-8008 Zürich, Switzerland
ivan.delchev@sap.com

ABSTRACT

Smart products are heterogeneous real-world objects with embedded computing and networking functionality. They are associated with digital object memories, which contain object-related product life-cycle information. Since smart products are in general not able to locally store all data objects of the digital object memory gathered during their life-cycle, there is a need for scalable data management mechanisms that achieve ubiquitous data availability. As part of a distributed storage mechanism, this paper proposes SORPA, a distributed and self-organizing replication algorithm for smart products. SORPA is capable of maintaining the locality of data objects and approaches the trade-off between data availability and consistency. In this way, SORPA fosters ubiquitous data availability, while taking into account the challenges of smart products.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*distributed systems, information networks*

Keywords

Ubiquitous Computing, Internet of Things, Smart Products, Distributed Storage, Data Replication, Self-Organization

1. INTRODUCTION

Since Mark Weiser's well-known visionary article "The Computer of the 21st Century", Ubiquitous Computing (UC) has received considerable research attention both in academia and industry [21]. UC envisions billions of intelligent physical objects ranging from objects equipped with smart labels (e.g., RFID tags) to *smart products* with embedded computing and networking capabilities. While smart labels are merely capable of being actively or passively read by interrogators, smart products are able to cooperate in distributed and dynamic environments during their entire product life-cycle [11].[1] Recent research activities in UC focus on the management of data objects taking into consideration the heterogeneity and resource limitation of intelligent physical objects as well as the scalability challenges of UC. Beside data modeling, data mining, and data security, this includes concepts for efficiently storing and distributing data objects, the management of obsolete data objects, as well as corresponding data availability and consistency requirements [7].

Digital object memories (DOM) associate object-related information with smart products across their entire product life-cycle. This includes design information such as blueprints, shipping information, information about performed maintenance operations, or usage history data [12]. Application scenarios that are built upon DOM require data management facilities tailored to the characteristics of smart products. They require data objects of the DOM to be ubiquitously available during the entire product life-cycle, regardless of the actual capabilities and environmental conditions of smart products. *Ubiquitous data availability* means that data is (i) available when needed and (ii) accessible with low latency, while (iii) meeting the expected level of consistency. For example, Smart Products as a Service (SPaaS) application scenarios require "real-time" access to data objects of the DOM in order to enable services such as installed base management or preventive maintenance. Data objects that are only accessible with high latency, outdated, or even not available at all would significantly affect the quality of SPaaS service offerings [3].

Smart products face several challenges that impede the achievement of ubiquitous availability of data objects. In addition to the above-mentioned heterogeneity and scalability challenges, smart products are typically *resource-constrained* with respect to their storage and communication capabilities. For this reason, smart products are in general not able to locally store all data objects of the DOM generated and required during their product life-cycle. On the other hand, not all smart products are able to store data in back-end systems at all times, because they might only support short-range communication technologies or (temporarily) reside in no-signal areas (i.e., end-to-end connectivity cannot be generally assumed) [10]. The *dynamics* of smart products poses another central challenge, which results from their mobility and potential intermittent connectivity. This requires

[1] In the remainder of this paper, the term *smart product* is used as an umbrella term covering all classes of intelligent physical objects.

smart products to be able to self-organize and cooperate in dynamic environments, and to cope with unpredictable communication delays [7].

This paper addresses the data management challenges of achieving ubiquitous availability of DOM. The contribution of this paper is an optimistic distributed storage mechanism that facilitates smart products to store data objects on-board and remotely on back-end systems. Even further, the mechanism enables smart products to store information on other smart products in the environment (in-network storage), thereby taking into account and advantage of the heterogeneity of smart products. To achieve ubiquitous availability of data objects, this distributed storage mechanism includes SORPA, a distributed and *self-organizing data replication algorithm* tailored to the challenges of smart products. While replication is a well-known means for enhancing data availability, it negatively impacts consistency of data objects and increases complexity of the related consistency model (cp. CAP theorem [5, 20]). By leveraging product life-cycle information, annotated workflows/business processes, and context information, SORPA replicates and places data objects in areas where they will likely be accessed in the nearby future. In this way, SORPA is able to maintain the locality of data objects and approaches the trade-off between availability and consistency. It accounts for the challenges of smart products and fosters ubiquitous data availability. Moreover, SORPA allows for disposing/outsourcing obsolete information. For example, information required in the manufacturing phase such as design instructions or bill of materials are neither needed in the subsequent use phase nor are they allowed to be accessed by end users. Such data objects should be either moved to smart products that possess high storage capacity or outsourced to back-end systems.

To cope with the dynamics of smart products, SORPA maintains a multi-level community-based overlay network. This overlay network takes into consideration capabilities and purpose of smart products and reflects their inherent heterogeneity. In this way, it enables smart products to self-organize (bottom-up integration) as well as to immediately communicate with and benefit from other smart products in their environment.

The remainder of the paper is structured as follows. Section 2 provides an overview of related work of both storage infrastructures for DOM and data replication algorithms. Thereafter, Section 3 presents a distributed storage concept for DOM. In addition to a system model (see Section 3.1), this includes the main contribution of the paper, the self-organizing data replication algorithm SORPA (see Section 3.2). The paper concludes in Section 4 with an outlook on future work.

2. RELATED WORK

According to [15], optimistic distributed storage concepts are characterized by a set of design decisions such as number of writers, operation maintenance and conflict management, as well as consistency maintenance and replication strategies. Recent research results have disclosed that smart products would benefit from a generic optimistic distributed storage mechanism that is not limited to DOM. Instead, it should further be applicable to other data objects such as user profiles, product life-cycle models, or workflows. According to the characteristics of UC, a distributed storage mechanism for smart products should support multiple writers and provide adequate data replication functionality to meet the availability requirements of different data classes.

The distributed mobile relational database Bayou [18] and Amazon's highly scalable key/value store Dynamo [6] are two well-known systems that sacrifice consistency in order to achieve high data availability. They both support multiple writers and facilitate the integration of application-specific semantically-enriched reconciliation logic. While Bayou only supports eventual consistency, Dynamo employs a configurable quorum-like approach to maintain replica consistency. Yet, they do not tackle the resource limitation of smart products. In addition, they both consider storage capacity as the only dimension of heterogeneity. Smart products, however, further vary in their communication and computation capabilities. For this reason, the complexity of distributed storage mechanisms meant for being deployed on a server landscape might significantly restrain their applicability to smart products systems.

The recently proposed object memory service (OMS) is a distributed storage infrastructure for DOM [16]. OMS combines on-board storage of resource-constrained products with off-board storage provided by the environment. For this purpose, a central resource directory is used to link products with data objects that are distributed in the environment. However, OMS does not support in-network storage as defined in Section 1, which is essential for achieving ubiquitous data availability. It only employs the limited on-board storage of products to place data objects with high availability requirements, while other information can solely be retrieved via the central resource directory. This can result in data objects being less or even not available in case they cannot be stored locally on a product that is not able to access the central resource directory (see challenges outlined in Section 1).

Replication strategies are part of most distributed storage mechanisms. Replication strategies are employed in Content Distribution Networks (CDN) to reduce access delay of client requests and network bandwidth consumption, as well as for balancing the load among replica servers. Opportunistic networks make use of data replication in order to distribute data objects between intermittently connected nodes according to the store-carry-and-forward paradigm. Finally, Peer-to-Peer (P2P) content distribution systems apply replication strategies to enhance availability of data objects.

In smart products systems, the placement of replicas as well as the optimization of the trade-off between availability and consistency have to be dealt with in order to achieve ubiquitous data availability. There exist several algorithms for dynamic placement and replacement of data objects such as the Top-K Most Frequently Requested algorithm proposed by [8], which aims at achieving high data availability in community-based P2P content distribution systems. However, within the knowledge of the authors, there are only two algorithms that integrate replication and consistency maintenance functionality.

MDCDN, a replication algorithm for mobile dynamic CDNs, aims at minimizing the overall network traffic. In addition to cost of replication operations and cost of indirect requests (i.e., requests that cannot be served locally but have to be forwarded), this includes cost of operations required to maintain the consistency of replicated data objects [2]. However, MDCDN is used in CDNs and does not cope with the heterogeneity and high dynamics of smart products. The IRM mechanism also integrates replication and consistency maintenance functionality. By relying on the file query and file update rate, which are autonomously measured by each node, IRM achieves a high utilization of replicas. This leads to a high level of data availability and reduces the consistency maintenance overhead [17]. However, even though IRM explicitly tackles and optimizes the trade-off between availability and consistency and copes with highly dynamic systems, its pure reactive concept is not fully suitable for smart products. Instead, a replication algorithm for smart products should make use of product life-cycle information and annotated workflows/business processes to *proactively* place replicas in areas where they will likely be required in the nearby future.

Finally, multi-level community-based network overlay structures have been investigated in different domains. NICE [4] and ZigZag [19] both maintain balanced application-layer multicast trees for streaming multimedia content to a large number of receivers. However, they both assume network homogeneity as well as end-to-end connectivity; two assumptions that are not given in smart products systems. Moreover, they rely on a central rendezvous point and apply a top-down maintenance strategy. Wireless sensor networks also make use of community-based network overlay structures. However, within the knowledge of the authors, most existing concepts are limited to a two-level structure or apply top-down maintenance strategies [1, 13]. SORPA, in contrast, is fully distributed and applies a bottom-up maintenance strategy based on node capability and purpose metrics.

3. DISTRIBUTED STORAGE

3.1 System Model

A distributed storage mechanism for smart products, which aims at achieving ubiquitous availability of data objects, requires a suitable system model and an overlay network architecture that accounts for the specifics of smart products. For scalability reasons, this model should enable smart products to autonomously self-organize in order to benefit from other smart products in their environment. This paper proposes an overlay network architecture that reflects the heterogeneity, resource limitation, and dynamics of smart products; a hierarchical, community-based overlay network that is built on top of the publish/subscribe communication paradigm. The multi-level structure of this pub/sub overlay network including the related entities, roles, and relations is depicted in Figure 1.

On the lowest level, smart products are clustered in *communities* by taking into consideration their capabilities, location, and purpose. This ensures that all smart products of a community can exchange information among each other and that they fulfill similar tasks with similar data needs. Each community consists of a set of *nodes* (i.e., smart products)

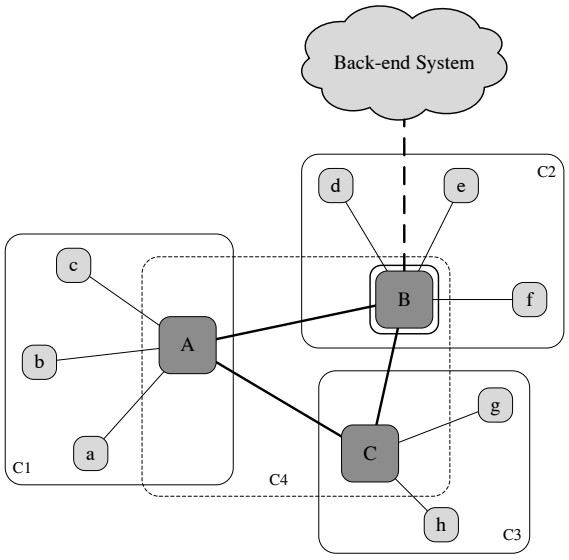

Figure 1: Hierarchical Pub/Sub Overlay Network

plus exactly one *supernode*. Supernodes are responsible for synchronizing divergent versions of data objects within their community. Furthermore, they maintain an index of data objects provided by nodes of their community, thus improving the efficiency of search requests. All smart products may play the role of a supernode, which is dynamically assigned based on capability metrics (e.g., storage, communication, energy). According to the hierarchical structure of the proposed pub/sub overlay network, supernodes that are able to communicate with each other are clustered in higher-level communities. Again, in these higher-level communities, all nodes but one represent nodes even though they played the role of supernodes in their lower-level communities. As opposed to related work outlined in Section 2, supernodes of higher-level communities can also be played by smart products that do not participate in any lower-level community.

The hierarchical structure of the proposed pub/sub overlay network is exemplified in Figure 1. Each of the three communities $C1$, $C2$, and $C3$ consists of a set of nodes (e.g., a, b, and c in community $C1$) as well as exactly one supernode (e.g., A in community $C1$). On the next level, the three supernodes A, B, and C are again clustered in the community $C4$, whereby B is dynamically elected as the community's supernode based on its capabilities.

As a real-world example, imagine a ship loaded with multiple containers with each container being loaded with goods. The proposed hierarchical network structure reflects the way the ship is loaded. Each container represents a community with one supernode, the container, and multiple child nodes, the different goods placed in the container. On the upper level, there is the ship community with the ship (or a dynamically elected container) as the supernode, and the containers loaded on the ship as the ship's children. Even further, one could think about another higher-level community, which contains multiple ships that are connected to a harbor or a logistics system as the community's supernode.

Communication links between nodes are modeled as publish/subscribe channels. Each node has a private channel with the node as the only consumer, which is used to directly address the node. In addition, each community consists of at least one public channel, which can for instance be used to propagate updated versions of data objects or to distribute lookup queries. Finally, there is a community-spanning rendezvous channel per level to which all supernodes of this level are subscribed. This channel is used as a rendezvous point by nodes joining a smart products system. To ensure the scalability of the proposed pub/sub overlay network and to avoid extensive reorganizations of its multi-level structure, reorganizations are merely performed in regular periods and limited to single communities. Otherwise, a very powerful smart product joining a smart products system could result in a reorganization of the entire pub/sub overlay network given that the role of a supernode is immediately re-assigned based on node capability metrics.

The hierarchical structure ensures that newly introduced smart products can self-organize into a community of the smart products system. This bottom-up integration allows smart products to immediately communicate with and benefit from other smart products in their environment. As an example, a joining smart product may directly make use of its supernode to outsource data that is highly required but that cannot be stored locally due to limited storage capacity. Finally, as stated by [9], a hierarchical clustering of smart products significantly enhances the scalability of smart products systems as well as the proposed distributed replication algorithm SORPA.

By fully relying on the publish/subscribe communication paradigm, SORPA does not depend on a concrete P2P overlay network implementation. Instead, the clear separation of concerns enables SORPA and the underlying P2P overlay network as well as the related routing and locating mechanisms to be optimized independently. Eventually, SORPA provides an easy-to-use and purpose-driven communication abstraction and can be integrated with all middlewares that comply with the publish/subscribe communication paradigm.

3.2 SORPA

As already outlined in Section 1, SORPA is embedded into an optimistic distributed storage mechanism for smart products. For reasons of completeness, it is assumed that the distributed storage module is decorated by a security component and makes use of a publish/subscribe communication middleware. Figure 2 presents the compositional structure of this mechanism in FMC Block Diagram notation.

The central component of the distributed storage module is the *distribution* component. It provides the external interface and orchestrates the other components of the module. The *storage* component encapsulates the local storage of a smart product (e.g., a relational database, a key/value store, or a triple store for semantic data), which is logically partitioned into (i) a private storage used exclusively by the smart product and (ii) a shared storage used to realize a shared memory within a community. This allows smart products to outsource data objects, i.e., to store data in-network. Furthermore, the shared storage of smart products is used by SORPA to distribute replicas of data objects in order to enhance data availability. The consistency maintenance functionality is provided by the *synchronization* component. As depicted in Figure 2, the proposed architecture facilitates the integration of multiple synchronization strategies. Due to the generic nature of the distributed storage module, the component performs syntactic reconciliation of divergent replicas and resolves conflicts based on pre-defined rules. Finally, the *pub/sub dispatcher* is responsible for de-multiplexing messages received from the publish/subscribe communication middleware. As an example, if a quorum-like consistency protocol is employed, then the pub/sub dispatcher will wait for N success responses before it notifies the distribution component that the operation has succeeded.

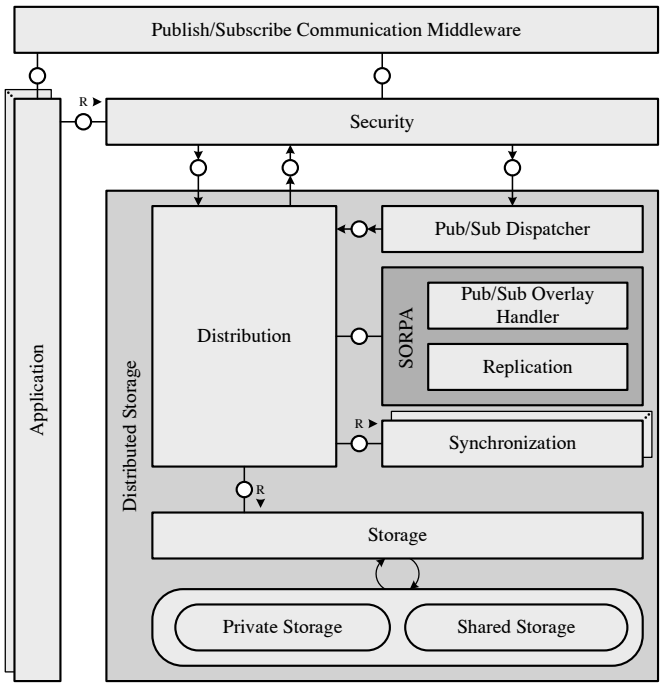

Figure 2: Distributed Storage - Architecture View

The dynamic structure of the distributed storage module is exemplified by a locally deployed application requesting a data object. The application sends its request to the distribution component via the security module, which may filter the request based on defined access rules. The distribution component queries the storage component with the given data request. In case the requested information is available locally, it is returned to the application. Otherwise, the request is dispatched into the smart products network by the distribution component. Given the network architecture presented in Section 3.1, this request is sent to the community's supernode, which either returns the private channel of the smart product storing the requested information based on a local index or dispatches the request to its own supernode (i.e., the request is dispatched to a higher-level community). Depending on the applied consistency protocol, the pub/sub dispatcher waits for multiple responses that are synchronized before the actual result is returned to the requesting application. Either way, the location of data ob-

jects as well as potential reconciliation processes are fully transparent to the requesting application.

With respect to the overall objective of achieving ubiquitous data availability across the entire product life-cycle, SORPA represents the central component of the distributed storage module. As depicted in Figure 2, it consists of two components: The *pub/sub overlay hander*, which has the task to maintain the pub/sub overlay network presented in Section 3.1, and the *replication* component that provides the actual replication strategy.

The *pub/sub overlay handler* provides the self-organization capability of SORPA. This functionality is presented by a smart product joining a smart products system. To self-organize into the smart products system, the joining smart product publishes an announcement message to the rendezvous channel described in Section 3.1. This message includes the identifier of the node as well as the node's profile (capabilities, purpose) and private channel. After having dispatched the announcement message, the joining node collects all responses from supernodes it receives within a defined time frame. Based on the metrics described before, it registers with the most suitable supernode as a child node (also in case this supernode is less capable than itself). If the joining node does not receive any response within the given time frame, it establishes a new community in which it plays the role of a supernode.

In order to maintain the pub/sub overlay network, the pub/sub overlay handler of each smart product playing the role of a child periodically publishes maintenance messages to the registered supernode. This maintenance message contains the current node profile as well as an index of the data stored on-board. To reduce the overhead of the maintenance process, both the node profile and the index are reduced to the changes performed since the last period (e.g., available storage capacity). The supernode aggregates these indices and compares each received node profile with its own status. In case the supernode recognizes that at least one of its children is more powerful than itself, it notifies the most powerful of its children. This notification includes the aggregated index and a list of all of its children. After receiving this message, the newly assigned supernode registers all other nodes of the community as children and publishes an overlay update message on the community's public channel. This enables all children to adapt to the reorganized community structure.

For scalability reasons, this reorganization is not directly propagated to the higher-level community the former supernode was part of. Instead, the new supernode joins the higher-level community as a child according to the above-described procedure. This is even the case, if the supernode of the higher-level community is the supernode that has just been replaced by the joining node. In addition to an enhanced scalability, this clear scoping further avoids collisions in the reorganization process. Finally, in case a supernode becomes overloaded, it is able to split its community. For this purpose, it elects the second best node of its community as a supernode of a new community and hands over a subset of its children to this new community.

The *replication* component of SORPA operates on the above-described pub/sub overlay network. It has the task to generate and place a sufficient number of replicas of data objects in order to achieve ubiquitous data availability. According to [14], the placement of data objects has a more significant effect on data availability than the number of replicas. This statement is further supported by [17], which aims at achieving a high utilization of replicas in order to optimize the trade-off between availability and consistency.

These concepts are adopted by SORPA. The placement of data objects is further enhanced by annotated models. This includes workflows/business processes annotated with data needs as well as product life-cycle models that comprises data requirements of each life-cycle phase. These models allow smart products to replicate and place data objects in areas where they will likely be accessed in the nearby future. As an example, let's assume a smart product A that executes the distributed workflow $WF1$, which is annotated with data needs for each task (e.g., executable code, configuration files, additional information). To proactively provide the smart products involved in $WF1$ with the required data and achieve low-latency data access, A sends the data needs of $WF1$ to its supernode B. Eventually, B leverages the multi-level overlay network in order to collect and distribute the required data to areas where they are needed to execute $WF1$.

This *proactive* replication is complemented with a monitoring-based replication mechanism. Similar to IRM, SORPA monitors the utilization of data objects stored on-board. On the one hand, SORPA can trigger the replication of highly utilized data objects to enhance availability of this data object or preserve the providing node from being overloaded. In this case, the replica is delivered to the node's supernode, which determines a node of its community that finally stores this new data object in its shared storage. On the other hand, SORPA can outsource or dispose unused data objects in order to free storage capacity. The replacement strategy applied by SORPA takes into consideration the availability requirement assigned to a data class and employs the well-known concept *least recently used*.

4. CONCLUSION & FUTURE RESEARCH

Digital object memories (DOM) associate object-related information with smart products accross their entire product life-cycle. Due to the resource limitation of smart products it cannot be generally assumed that all data objects of the DOM, which are gathered during the product life-cycle, can be stored locally on the product. Yet, to achieve the full potential of the DOM, it is invaluable to make it available across the entire product life-cycle.

This paper proposes SORPA, a distributed and self-organizing replication algorithm for smart products. SORPA aims at achieving ubiquitous data availability during the entire product life-cycle. This means, data is available when needed and accessible with low latency, while meeting the expected level of consistency. SORPA is integrated into an optimistic distributed storage mechanism, which enables smart products to store data objects locally on the product, remotely on back-end systems, as well as on other nearby smart products (in-network storage).

SORPA maintains a hierarchical, community-based publish/subscribe overlay network, which takes into consideration capabilities and purpose of smart products and reflects their inherent heterogeneity. It optimizes the placement of replicas and approaches the trade-off between data availability and consistency. For this purpose, SORPA takes into consideration the utilization of replicas, annotated workflow/business process and product life-cycle models, as well as availability requirements of data classes. This leads to a reduction of the overall number of replicas plus a higher replica utilization. Eventually, this results in a smaller consistency maintenance overhead without sacrifycing data availability.

Future research includes the further development of SORPA, focussing in particular on the self-organization functionality and the combined proactive and monitoring-based replication strategy. Moreover, the investigation and setup of a simulation model for smart products as well as the definition of evaluation metrics and test scenarios are subject for future research. This provides the basis for evaluating different replication concepts as well as the performance and scalability of the proposed publish/subscribe overlay network. Finally, future studies will cover data management issues in selected application scenarios such as the question of whether and how to aggregate data objects within different levels of the proposed hierarchal structure of smart products.

Acknowledgments

Part of this research has been funded under the EC 7th Framework Programme, in the context of the SmartProducts project (231204).

5. REFERENCES

[1] A. A. Abbasi and M. Younis. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, 30(14-15):2826–2841, 2007.

[2] W. M. Aioffi, G. R. Mateus, J. M. Almeida, and D. S. Mendes. Mobile dynamic content distribution networks. *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 87–94, 2004.

[3] G. Allmendinger and R. Lombreglia. Four strategies for the age of smart services. *Harvard business review*, 83(10):1–11, Oct. 2005.

[4] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 205–217, Pittsburgh, Pennsylvania, USA, 2002. ACM.

[5] E. Brewer. Towards robust distributed systems. In *PODC '00: Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing*, volume 19, page 7, Portland, Oregon, United States, 2000. ACM New York, NY, USA.

[6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. In *SOSP '07: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles*, pages 205–220, New York, NY, USA, 2007. ACM.

[7] M. J. Franklin. Challenges in Ubiquitous Data Management. In *Informatics - 10 Years Back. 10 Years Ahead.*, pages 24–33, London, UK, 2001. Springer-Verlag.

[8] J. Kangasharju, K. Ross, and D. Turner. Optimizing File Availability in Peer-to-Peer Content Distribution. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pages 1973–1981. IEEE, 2007.

[9] M. Karlsson, C. Karamanolis, and M. Mahalingam. A Framework for Evaluating Replica Placement Algorithms, 2002.

[10] M. Miche, D. Schreiber, and M. Hartmann. Core Services for Smart Products. In *AmI-Blocks'09: 3rd European Workshop on Smart Products*, pages 1–4, Salzburg, 2009.

[11] M. Mühlhäuser and I. Gurevych, editors. *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*. IGI Publishing, Hershey PA, USA, 2007.

[12] J. Neidig and P. Stephan. An Object Memory Modeling Approach for Product Life Cycle Applications. In *Proceedings of the 5th International Conference on Intelligent Environments. 1st Workshop on Digital Object Memories*. IOS Press, 2009.

[13] B. C. Okeke and K. L. E. Law. Multi-level Clustering Architecture and Protocol Designs for Wireless Sensor Networks. In *WICON '08: Proceedings of the 4th Annual International Conference on Wireless Internet*, pages 1–9, Maui, Hawaii, 2008. ICST, Brussels, Belgium, Belgium.

[14] G. On. *Quality of Availability for Widely Distributed and Replicated Content Stores*. Phd thesis, Technische Universität Darmstadt, 2004.

[15] Y. Saito and M. Shapiro. Optimistic Replication. *ACM Computing Surveys*, 37(1):42–81, 2005.

[16] M. Schneider. Towards a general object memory. *UbiComp 2007 Workshop Proceedings*, 2007.

[17] H. H. Shen. IRM : Integrated File Replication and Consistency Maintenance in P2P Systems. *IEEE Transactions on Parallel and Distributed Systems*, 21(1):100–113, 2010.

[18] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in Bayou, a weakly connected replicated storage system. In *Proceedings of the 15th ACM symposium on operating systems principles*, pages 172–182. ACM, 1995.

[19] D. Tran, K. Hua, and T. Do. A Peer-to-Peer Architecture for Media Streaming. *IEEE Journal on Selected Areas in Communications*, 22(1):121–133, 2004.

[20] W. Vogels. Eventually Consistent. *Queue*, 6(6):14—-19, 2008.

[21] M. Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3:3–11, 1999.