ON THE STUDY OF DYNAMIC AND ADAPTIVE DEPENDABLE DISTRIBUTED SYSTEMS

J. E. Armendáriz-Iñigo, J. R. Juárez-Rodríguez, J. R. González de Mendívil
Universidad Pública de Navarra, 31006 Pamplona, Spain

F. D. Muñoz-Escoí, R. de Juan-Marín
Instituto Tecnológico de Informática, Universidad Politécnica de Valencia, 46022 Valencia, Spain

Keywords: Distributed systems, Availability, Dependability, Dynamic systems, Data consistency.

Abstract: Due to the usage of MANETs and some kinds of collaborative applications (P2P), current distributed systems are becoming increasingly dynamic; i.e., it is difficult to manage membership information and to forecast the accessibility of each system node. Moreover, dependable applications for static distributed systems also need to provide good adaptability levels (to different request arrival rates, usage patterns, classes of requests,...) and good scalability; a case to study is the cloud computing paradigm. Development of dependable applications in dynamic and adaptive systems is not trivial, since both dynamism and adaptability may compromise algorithm liveness or may complicate the design of such algorithms, specially those best suited for static systems. Strategies for building adaptable and scalable dependable services (based on "cloud systems") will be surveyed and improved. Moreover, an efficient support for dependable applications in dynamic systems will be provided, combining three different approaches: relaxed consistency models, interconnection protocols (for supporting both consistency and multicasting) and reconciliation strategies. Last but not least, also the usage and support for integrity constraints in replicated systems will be analyzed and improved for dynamic systems.

1 INTRODUCTION

In the last decade, there have been several advances in the distributed systems field that have allowed the creation of new decentralized applications. A first sample are P2P applications, e.g. Napster, as a first system of this kind, that still used a centralized directory, but where all data transfers were made between peers that collaborated in order to share files. Later on, the need of a centralized service is skipped, such as Gnutella, although it raised difficulties in order to find the files to be shared, since search messages need to be flooded into the network. This compromised system scalability, since this excessively increased the amount of messages needed for obtaining the searched file and did not guarantee that such file could be located, even when multiple copies of such file did exist in the system. Another evolution of these P2P systems were their structured variants, e.g. Tapestry (Zhao et al., 2004), providing something similar to a distributed hash table (DHT). In this case, such system organization or "structure" allows a fast localization of any shared content, guaranteeing system scalability. However, structured variants also raise serious problems when their nodes join and leave the system very often. Note that these P2P applications are a valid example of dynamic distributed system, since none of their nodes needs to know all the other potential collaborators system nodes. Moreover, P2P system nodes do not remain in the system for long time intervals, causing system membership to be highly dynamic.

As it has been shown in the previous paragraph, system decentralization may lead to obtaining dynamic distributed systems. Another example of dynamic system is one consisting of a set of mobile computers interconnected through a wireless network. As in the previous case, the usage of such systems has become popular nowadays. Currently, wireless networks can be found in many environments (universities, enterprises, and airports, to name a few, or even at home) since many users prefer a com-

E. Armendáriz-Iñigo J., R. Juárez-Rodríguez J., R. Gonzalez de Mendivil J., D. Muñoz-Escoí F. and de Juan-Marín R. (2009).
ON THE STUDY OF DYNAMIC AND ADAPTIVE DEPENDABLE DISTRIBUTED SYSTEMS.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 183-186
DOI: 10.5220/0002279801830186
Copyright © SciTePress

puter network of this kind. In this kind of systems, sensor networks (mainly oriented towards monitoring applications in different environments: weather forecasting, zoological surveys (habitat-oriented), domotics, biomedical applications, etc.) and VANETs (MANETs whose nodes are plugged into vehicles, providing a complementary set of services to the vehicles' driver or users) can also be included. All these cases are samples of dynamic distributed systems. Applications developed on them need to consider a system model that is not equivalent to the traditional one. Besides assuming asynchrony, in these cases the application designer is unable to know which is the actual system membership (each node does only interact with a small part of the system population), and he/she should also assume that the probability of node failure or disconnection is high. Thus, the design of efficient distributed algorithms in these environments is a challenging task that has deserved attention in the last years. So, several theoretical results have been published in this area. For instance, (Mostéfaoui et al., 2005) already identified these problems and proposed some parameters and basic primitives that allow algorithm migration from static to dynamic systems. Their proposal uses an alpha parameter, consisting in the number of nodes that could be considered as a stable core in such dynamic system (such core does not need to be perpetual; its nodes should be alive and active in a time interval long enough to ensure algorithm progress), and several communication primitives (a termination-guaranteed query-response, that is reactivated as soon as alpha replies are collected, and a reliable and persistent multicast; i.e., that guarantees message delivery to the nodes that have joined the system whilst the message was multicast). Using these tools, such paper proves that a leader-election algorithm can be migrated to a dynamic system, despite being a problem that has traditionally been considered unresolvable when system nodes' identifiers were unknown. A second proposal was presented in (Baldoni et al., 2007) where different characteristics of several dynamic systems are surveyed and a first classification is provided, depending on the basis provided for designing distributed algorithms. This proves that this is an interesting research line, since it will allow algorithm migration (or prove that such migration is not possible in some cases) from static systems to dynamic ones. Thus, our aim consists in choosing several problems already solved in static systems (consensus, leader election, mutual exclusion, global state collection, ...) and to analyze in which dynamic settings such problems could still be solved. Note that it has been widely accepted that a distributed system should be able to overcome its node and communication failures in order to be useful. Due to this, it is common that the applications being developed on these system assume some kind of dependable support (or even that the underlying system is dependable), but it is not trivial to provide such dependable support in a dynamic system. Another objective is to improve the dependability of applications designed, implemented and deployed in this kind of systems.

On the other hand, systems that consist of a stable set of nodes able to adapt themselves to different kinds of load or to different kinds of environments can also be classified as dynamic. Thus, in the field of data replication management it has been provided a meta-protocol (Ruiz-Fuertes et al., 2007) that may simultaneously support multiple replication protocols, allowing that each application uses a different replication protocol in order to manage its transactions, improving its performance and reducing the abort rates of those transactions that could use a relaxed isolation level (note that each protocol may support a different isolation level, if needed). Note also that different database replication protocols provide different performance when they manage different load levels, as proven by (Wiesmann and Schiper, 2005). Thus, this meta-protocol is complemented by a system load monitor, could allow that its replication middleware chooses the best protocol for each transaction according to the current system load configuration when such transaction is started, improving system adaptability.

This adaptability to variable loads, and the cost reduction implied by hardware downscaling, have given rise to a new distributed computing paradigm called "cloud computing". The aim of these systems consists in enabling possibly huge quantities of networked computing resources to be "contracted" in order to achieve the execution of large-scale applications of enterprise customers. To their clients, the contracted resources become "remote", being accessed via the Internet (the virtual "cloud" location which gave name to this new paradigm). Thus, the IT departments of cloud computing clients only need to worry about developing their applications (usually, web services), and deploying them in the hired virtual computers. They do not need to purchase and operate a proprietary computing center, such that big investments and administration costs are avoided. On the other hand, the company providing such clouds must maintain enough virtual computing capacities for guaranteeing the contracted services and for ensuring easy adaptation to different request loads (note that the final user of such services could be a third company or the users of the services provided by the client company). There are multiple problems that need to be solved in this kind of cloud systems: (1) scalability, (2) availability, (3) security, (4) virtualisation (hired computers are only virtual nodes whose load will be supported by actual computers; this allows an easy isolation of each virtual node, ensuring a secure deployment, but it also complicates a bit the overall system management regarding the other requirements: scalability and availability). Finally, client companies will be budgeted with a static payment (depending on the bandwidth and minimal computing power that have been contracted), plus a variable amount that depends on the actual resource usage. There are several proposals for this computing model: Amazon Elastic Compute Cloud (EC2) (Amazon, 2009), Google App Engine (Google, 2009), Windows Azure (Microsoft, 2009), ... This confirms that service scalability, adaptability, security and availability are important issues that need additional research. Solutions being described in the white papers referenced above are not complex, but it seems appropriate to propose advanced solutions to these problems, and this is our main objective in this proposal. We can not develop an entire cloud system, since this demands a lot of time and effort (and the solutions developed by Amazon, Google, Microsoft, IBM and other companies will have been highly improved in the meantime). So, our objective will be centered in proposing complementary solutions that could be easily included in such commercial products or that could be directly used by the client companies discussed above.

2 MAIN RESEARCH GOALS

As already mentioned in the introduction, there are several types of dynamic distributed systems and all of them have received special coverage in the last years. However, and with a few exceptions (Mostéfaoui et al., 2005; Baldoni et al., 2007), there have not been any theoretical surveys about how to adapt the classical solutions for static systems to these new environments, and about how to formally characterize the latter. Thus, there are many things of interest in this research line, from both theoretical and practical points of view. Every application running in a dynamic system shares some amount of information among its system nodes. This implies that the consistency protocols already developed for static systems could be improved and adapted for these dynamic environments, where node connectivity cannot be always guaranteed. To this end, an initial basis could be provided by interconnection protocols for communication systems (Álvarez et al., 2008) (needed for supporting some consistency model when previously isolated node groups are re-joined, a frequent event in a dynamic system) or some reconciliation protocols (Asplund et al., 2007) that have been recently published. This need of scalability and adaptation to highly variable workloads has been identified by the most important companies in the field of web service technologies and operating systems, driving them to propose the cloud computing systems already discussed in the introduction. This shows that there are not good solutions (at least, demanding a low or moderate computing effort) for these problems and that additional research is needed in this area. Thus, some of their persistence mechanisms are still very basic (Amazon, 2009) (providing a shared-disk image where multiple copies of the information are stored) or they provide an interface very close to the web services (Google, 2009; Microsoft, 2009) but without relational support (losing thus general functionality).On the other hand, in order to improve scalability, the general principle (Helland and Campbell, 2009) consists in relaxing consistency while increasing the degree of asynchrony, using consistency and concurrency control mechanisms that are very optimistic. Such principle could be also used in a dynamic system, improving application performance, but relaxing the consistency provided to the users. These are going to be the main goals of our study:

Theoretical Study and Classification of Different Types of Dynamic Systems and Mechanisms Needed for Migrating Algorithms from Static to Dynamic Systems. Although there have been some results (Mostéfaoui et al., 2005; Baldoni et al., 2007) in this area, such classifications could still be refined and the support suggested to each identified class could also be extended, evaluating which kinds of algorithms designed for static systems could be migrated to a dynamic one.

Study, Design and Prototyping of Process and Persistent Data Management Strategies Providing Adaptability and Scalability. Regarding adaptability and scalability, the meta-protocol (Ruiz-Fuertes et al., 2007) that allows the coexistence of multiple replication protocols in the same middleware. On this support, it will be necessary to design and to implement a module of load and performance analyses that chooses at every moment the most suitable protocol for the current system characteristics. But we must extend such support to guarantee a greater scalability. To this end, it will be necessary to evaluate which replication strategies will be most appropriate, what type of transactional support could be provided and what consistency degree could be maintained. The proposed solutions in “cloud systems” use relaxed strategies in those three variants and they do not always consider replication.

Adaptation and Improvement of Existing Consistency Protocols for Static Systems in Order to use them in Dynamic Systems. Considering some existing results (Álvarez et al., 2008), it seems appropriate to select FIFO or causal consistency models in dynamic systems, thus allowing the usage of simple interconnection protocols for those subgroups that remained in isolation and are currently rejoining in order to compound a bigger system. Other solutions (Asplund et al., 2007) are based on the usage of reconciliation strategies, choosing which updates could be accepted and which others should be rejected or adapted. Interconnection and reconciliation strategies have evolved independently. Their combination has not yet been studied. More recently, it has been proposed the *eventually consistent* model for large scale distributed systems (Vogels, 2009). Our proposal tries to combine their best characteristics in those settings where such approach would make sense. Moreover, the theoretical results provided in the context of our first objective will also affect this study of consistency protocols.

Analysis, Design and Prototyping of a Dependable Middleware System for Dynamic Environments. The static solutions taken as a basis for dependable middleware development would be placed in modules that could be replaced by other appropriately tailored for dynamic settings. The resulting system should be able to provide a good support for both static and dynamic systems. Multiple mechanisms for ensuring security should be also supported by this middleware.

Extension and Dynamization of Integrity Support and Usage in Replicated Systems. It has been studied the integrity constraint management in replicated databases (Lin et al., 2009); However, another possible extension consists in including dynamic constraints (i.e., triggers) in our management, and also constraints of arbitrary generality. Up to now, only built-in constraints declared in the database schema have been maintained by the system. Another extension could be based on migrating our mechanisms to other kinds of replication protocols. For instance, (Asplund et al., 2007) analyzed the consistency-availability trade-off in partitionable systems, with a constraint-based consistency management. A last extension could consist in the evaluation and measurement of the resulting (in)consistency degree. Partial database replication is another field where our proposed mechanisms could be used. since partial replication is needed in dynamic systems with limited resources of computing power and storage capacity. There have not been any important results in this field, up to now.

ACKNOWLEDGEMENTS

This work has been supported by the Spanish MEC under research grant TIN2006-14738-C02.

REFERENCES

Álvarez, A., Arévalo, S., Cholvi, V., Fernández, A., and Jiménez, E. (2008). On the interconnection of message passing systems. *Inf. Process. Lett.*, 105(6):249–254.

Amazon (2009). Amazon elastic compute cloud (amazon ec2). Accessible in URL: http://aws.amazon.com/ec2/.

Asplund, M., Nadjm-Tehrani, S., Beyer, S., and Galdámez, P. (2007). Measuring availability in optimistic partition-tolerant systems with data constraints. In *DSN*, pages 656–665.

Baldoni, R., Bertier, M., Raynal, M., and Piergiovanni, S. T. (2007). Looking for a definition of dynamic distributed systems. In *PaCT*, volume 4671 of *LNCS*, pages 1–14. Springer.

Google (2009). What is google app engine? Accessible in URL: http://code.google.com/appengine/docs/whatisgoogleappengine.html.

Helland, P. and Campbell, D. (2009). Building on quicksand. In *CIDR*.

Lin, Y., Kemme, B., Patiño-Martínez, M., Jiménez-Peris, R., and Armendáriz-Iñigo, J. E. (2009). Snapshot isolation and integrity constraints in replicated databases. In *ACM TODS*. To appear.

Microsoft (2009). Azure services platform. Accessible in URL: http://www.microsoft.com/azure/default.mspx.

Mostéfaoui, A., Raynal, M., Travers, C., Patterson, S., Agrawal, D., and Abbadi, A. E. (2005). From static distributed systems to dynamic systems. In *SRDS*, pages 109–118.

Ruiz-Fuertes, M. I., de Juan-Marín, R., Pla-Civera, J., Castro-Company, F., and Muñoz-Escoí, F. D. (2007). A metaprotocol outline for database replication adaptability. In *OTM Workshops (2)*, volume 4806 of *LNCS*, pages 1052–1061. Springer.

Vogels, W. (2009). Eventually consistent. *Commun. ACM*, 52(1):40–44.

Wiesmann, M. and Schiper, A. (2005). Comparison of database replication techniques based on total order broadcast. *IEEE Trans. Knowl. Data Eng.*, 17(4):551–566.

Zhao, B. Y., Huang, L., Stribling, J., Rhea, S. C., Joseph, A. D., and Kubiatowicz, J. (2004). Tapestry: a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53.