P2P Data Replication: Techniques and Applications

Evjola Spaho[1], Admir Barolli[2], Fatos Xhafa[3], and Leonard Barolli[1]

[1] Fukuoka Institute of Technology,
3-30-1 Wajiro-Higashi, Higashi-Ku, Fukuoka 811-0295, Japan
evjolaspaho@hotmail.com, barolli@fit.ac.jp

[2] Hosei University,
3-7-2, Kajino-cho, Koganei-shi, Tokyo 184-8584, Japan
admir.barolli@gmail.com

[3] Technical University of Catalonia,
C/Jordi Girona 1-3, 08034 Barcelona, Spain
fatos@lsi.upc.edu

Abstract. Peer-to-Peer (P2P) computing systems offer many advantages of decentralized distributed systems but suffer from availability and reliability. In order to increase availability and reliability, data replication techniques are considered commonplace in P2P computing systems. Replication can be seen as a family of techniques. Full documents or just chunks can be replicated. Since the same data can be found at multiple peers, availability is assured in case of peer failure. Consistency is a challenge in replication systems that allow dynamic updates of replicas. Fundamental to any of them is the degree of replication (full vs. partial), as well as the source of the updates and the way updates are propagated in the system. Due to the various characteristics of distributed systems as well as system's and application's requirements, a variety of data replication techniques have been proposed in the distributed computing field. One important distributed computing paradigm is that of P2P systems, which distinguish for their large scale and unreliable nature. In this chapter we study some data replication techniques and requirements for different P2P applications. We identify several contexts and use cases where data replication can greatly support collaboration. This chapter will also discuss existing optimistic replication solutions and P2P replication strategies and analyze their advantages and disadvantages. We also propose and evaluate the performance of a fuzzy-based system for finding the best replication factor in a P2P network.

Keywords: P2P Systems, Data replication, Replication techniques, Data availability, P2P applications.

1 Introduction

Peer-to-peer (P2P) systems have become highly popular in recent times due to their great potential to scale and the lack of a central point of failure. Thus, P2P architectures will be important for future distributed systems and applications. In such systems, the computational burden of the system can be distributed to peer nodes of the system. Therefore, in decentralized systems users themselves become actors by sharing, contributing, and controlling the resources of the system. This characteristic makes P2P systems very interesting for the development of decentralized applications [1, 2].

Important features of such applications include the security, capability to be self-organized, decentralized, scalable, and sustainable [3–6]. P2P computing systems offer many advantages of decentralized distributed systems but suffer from availability and reliability. In order to increase availability and reliability, data replication techniques are considered commonplace in distributed computing systems [7–9]. Initial research work and development in P2P systems considered data replication techniques as means to ensure availability of static information (typically files) in P2P systems under highly dynamic nature of computing nodes in P2P systems. For instance, in P2P systems for music file sharing, the replication allows to find the desired file at several peers as well as to enable a faster download. In many P2P systems the files or documents are considered static or, if the change, new versions are uploaded at different peers. In a broader sense, however, the documents could change over time and thus, the issues of availability, consistency and scalability arise in P2P systems. Consider for instance, a group of peers that collaborate together in a project. They share documents among them, and, in order to increase availability, they decide to replicate the documents. Because documents can be changed by peers, for instance different peers can edit the same document, thus changes should be propagated and made to the replicas to ensure consistency. Moreover, the consistency should be addressed under the dynamics of the P2P systems, which implies that some updates could take place later (as a peer might be off at the time when document changes occurred). In this chapter we discuss different data replication techniques in P2P and different context and uses of data replication.

This chapter is organized as follows. In Section 2, we briefly describe the main characteristics of P2P systems. In Section 3, we explain different P2P data replication techniques. Section 4 describes replication requirements and solutions for different applications. In Section 5, we show our proposed fuzzy-based system for finding the best replication factor in a P2P network and give some simulation results. In Section 6, we give a brief discussion and analysis of replication in P2P. Section 7 concludes this chapter.

2 P2P Systems

A P2P system [10] is a self-organizing system of equal and autonomous entities, which aims for the shared usage of distributed resources in networked environment avoiding central services. A peer is an entity in the system, usually an application running on a device, or the user of such an application. All peers should be of equivalent importance to the system, no single peer should be critical to the functionality of the system.

In a P2P network, peers communicate directly with each other to exchange information. One particular example of this information exchange, that has been rather successful and has attracted considerable attention in the last years, is file sharing. These kind of systems are typically made up of millions of dynamic peers involved in the process of sharing and collaboration without relying in central authorities. P2P systems are characterized by being extremely decentralized and self-organized. These properties are essential in collaborative environments. The popularity and inherent features of these systems have motivated new research lines in the application of distributed P2P computing.

P2P applications such as distributed search applications, file sharing systems, distributed storage system and group ware have been proposed and developed [11], [12], [13], [14], [15].

Some of the essential features of P2P systems are:

– The peers should have autonomy and be able to decide services they wish to offer to other peers.
– Peers should be assumed to have temporary network addresses. They should be recognized and reachable even if their network address has changed.
– A peer can join and leave the system at its own disposal.

P2P systems have the following benefits.

– **Use of the previously unused resources:** On home and office computers, processing cycles are wasted constantly while the computer is on but underutilized/idle (generally overnight and during non-business hours). The disk storage is typically underutilized, as these computers are used mostly for simple nonintensive tasks. The P2P-based application can make use of these resources, thereby increasing utilization of an already paid resource.
– **Potential to scale:** The resources of the server or server-cluster limit the capabilities of the client-server system. As the number of clients increases, it becomes difficult to keep up with demand and maintain the performance and service at the required level. By distributing demand and load on the shared resources, the bottlenecks can be eliminated and a more reliable system achieved.
– **Self-organization:** P2P systems build and organize themselves. Each peer dynamically discovers other peers and builds the network. They organize according to their preferences and current conditions within the peer group. If a popular peer is overloaded and poor performance occurs, consumer peers can switch to another provider, effectively re-balancing load, and changing the network topology.

P2P computing systems offer many advantages of decentralized distributed systems but suffer from availability and reliability. In order to increase availability and reliability, data replication techniques are considered commonplace in P2P computing systems.

3 Data Replication and Update Management in P2P Systems

Initial research work and development in P2P systems considered data replication techniques as a means to ensure availability of static information (typically files) in P2P systems under highly dynamic nature of computing nodes in P2P systems. However, considering only static information and data might be insufficient for certain types of applications in which documents generated along application lifecycle can change over time. The need is then to efficiently replicate dynamic documents and data.

Data replication aims at increasing availability, reliability, and performance of data accesses by storing data redundantly [16–19]. A copy of a replicated data object is called a replica. Replication ensures that all replicas of one data object are automatically updated when one of its replicas is modified. Replication involves conflicting goals with respect to guaranteeing consistency, availability, and performance. Data replication techniques have been extensively used in distributed systems as an important effective mechanism for storage and access to distributed data. Data replication is the process of creating copies of data resources in a network. Data replication is not just copying data at multiple locations as it has to solve several issues. Data replication can improve the performance of a distributed system in several ways:

- **High availability, reliability, and fault tolerance:** Data replication means storing copies of the same data at multiple peers, thus improving availability and scalability. Full documents (or just chunks) can be replicated. Since the same data can be found at multiple peers, availability is assured in case of peer failure. Moreover, the throughput of the system is not affected in case of a scale-out as the operations with the same data are distributed across multiple peers.
- **Scalability:** Service capacity increased due to server load can be decreased. Response time and QoS requirements can be greatly improved.
- **Performance:** Increased performance due to data access.
- **"Fail Safe" infrastructures:** Replication is a choice for today's critical IT systems as replication is key to reducing the time for service recovery.

However, consistency is a challenge in replication systems that allow dynamic updates of replicas.

3.1 Data Replication Update Management

In [30], replica control mechanisms are classified using three criteria (see also [20], [21], [22]): where updates take place (single-master vs. multi-master), when updates are propagated to all replicas (synchronous vs. asynchronous) and how replicas are distributed over the network (full vs. partial replication) as shown in Fig. 1.

3.1.1 Where Updates Take Place: Single-Master and Multi-master

In the Single-master approach, there is only a single primary copy for each replicated object. The single-master allows only one site to have full control over the replica (read and write rights) while the other sites can only have a read right over the replica. This model is also known as the master-slave approach due to the interaction of the master node with the other nodes (slaves) storing the replica. Advantage of this model is the centralization of the updates at a single copy, simplifying the concurrency control. The disadvantage of this model is a single point of failure that can limit the data availability. The single-master approach is depicted in Fig. 2.

The updates can be propagated through *push mode* or *pull mode*. In *push mode*, it is the master that initiates the propagation of the updates, while in the case of *pull mode*, the slave queries the master for existing updates.

In the Multi-master approach, multiple sites hold primary copy of the same object. All these copies can concurrently updated. Multiple sites can modify their saved replicas. This approach is more flexible than single-master because in the case of one master failure, other masters can manage the replicas. The multi-master approach is presented in Fig. 3.

3.1.2 When Updates Are Propagated: Full Replication and Partial Replication

There are two basic approaches for replica placement: full replication and partial replication. Full replication takes place when each participating site stores a copy of every shared object. Every site should have the same memory capacities in order to replace any other site in case of failure. Figure 4 shows how two objects A and B respectively are replicated over three sites.

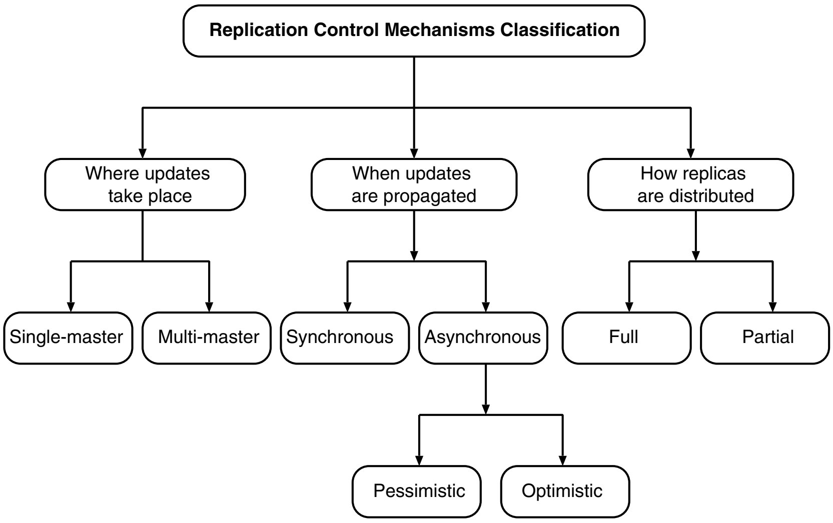


Fig. 1. Replica control mechanisms classification

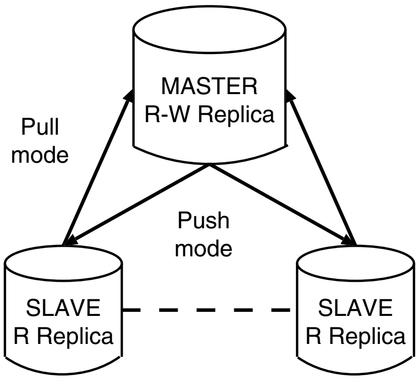


Fig. 2. Single-master replication

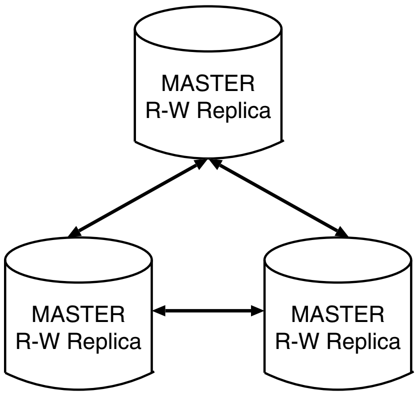


Fig. 3. Multi-master replication

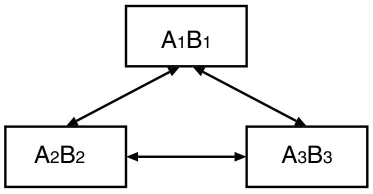


Fig. 4. Full replication with two objects A and B

In partial replication, each site holds a copy of a subset of shared objects so the sites can take different replica objects (see Fig. 5). This approach requires less storage space because updates are propagated only toward the affected sites. But this approach limits load balance possibilities as certain sites are not able to execute a particular type transaction [23]. In partial replication, it is important to

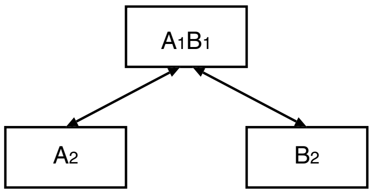


Fig. 5. Partial replication with two objects A and B

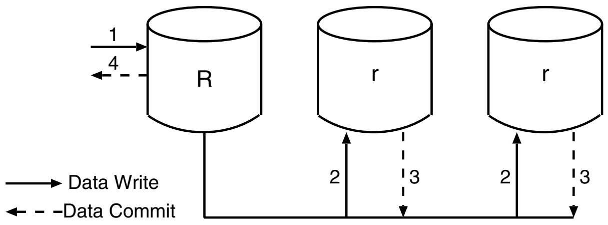


Fig. 6. Synchronous replication

find the right replication factor. Careful planning should be done when deciding which documents to replicate and at which peers.

3.1.3 How Replicas Are Distributed: Synchronous and Asynchronous Replication

Replication can be performed in an eager (synchronous) or lazy way (asynchronous) [30]. In the case of eager replication, when one replica is modified by a transaction, the other replicas of the concerned data object are updated within the original database transaction, as opposed to lazy replication where only the originally accessed replica is updated within the original transaction, while the other replicas are updated in separate transactions. The node that initiates the transaction propagates the update operations within the context of the transaction to all the other replicas before committing the transaction (see Fig. 6).

Combinations of synchronous and asynchronous replication have also been studied [31, 32]. In Synchronous replication, the node that initiate the transaction (set of update operations) propagates the update operations within the context of the transaction to all the other replicas before committing the transaction. There are several algorithms and protocols to achieve this behavior [33, 34]. Synchronous propagation enforces mutual consistency among replicas. In [33] authors define this consistency criteria as one-copy-serializability. The main advantage of synchronous propagation is to avoid divergences among replicas. The drawback is that the transaction has to update all the replicas before committing.

The asynchronous approach does not change all replicas within the context of the transaction that initiates the updates. The transaction is first committed at

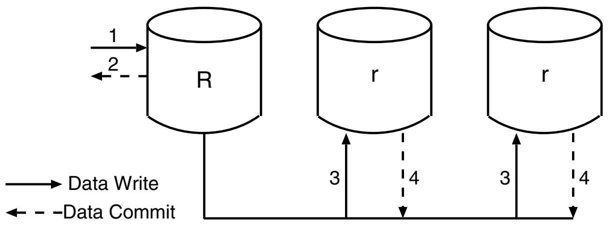


Fig. 7. Asynchronous replication

the local site and after that the updates are propagated to the remote sites as shown in Fig. 7. An advantage of asynchronous propagation is that the update does not block due to unavailable replicas, which improves data availability. The asynchronous replication technique can be classified as optimistic or non-optimistic in terms of conflicting updating [35, 36].

Pessimistic Approaches
These approaches combine lazy replication with one-copy-serializability. Each replicated data item is assigned a primary copy site and multiple secondary copy sites and only the primary copy can be modified. But because primary copies are distributes across the system, serializability cannot be always guaranteed [37]. In order to solve these problems, constraints on primary and secondary copy placements must be set. The problem is solved with the help of graph representation. The Data Placement Graph (DPG) is a graph where each node represents a site and there is a directed edge from Site i to Site j if there is at least one data item for which Site i is the primary site and Site j is the secondary site. The configurations a DPG can have for the system to be serializable is determined with the Global Serialization Graph (GSG). The GSG is obtained by taking the union of nodes and edges of the Local Serialization Graph (LSG) at each site. The LSG is a partial order over the operations of all transactions executed at that site. A DPG is serializable only if GSG is acyclic.

The above method can be enhanced so that it allows some cyclic configurations. In order to achieve this, the network must provide FIFO reliable multicast [35]. The time needed to multicast a message from one node to any other node is not greater than Max and the difference between any two local clocks is not higher than ε. Thus, how a site receives the propagated transaction in at most $Max + \varepsilon$ units of time, chronological, and total orderings can be assured without coordination among sites. The approach reaches the consistency level equivalent to one-copy-serializability for normal workloads and for bursty workloads it is quite close to it. The solution was extended to work in the context of partial replication too [23]. Pessimistic approaches have the disadvantage that two replicas might not be consistent for some time interval. That is why the criterion of consistency freshness is being used, which is defined as the distance between two replicas.

Optimistic Approaches
Optimistic approaches are used for sharing data efficiently in wide-area or mobile environments. The difference between optimistic and pessimistic replication is that the first does not use one-copy-serializability. Pessimistic replication use synchronization during replica propagation and block other users during an update. On the other hand, optimistic replication allows data to be accessed without using synchronization, based on the assumption that conflicts will occur only rarely, if at all. Update propagation is made in the background so that it is possible to exist divergences between replicas. Conflicting updates are reconciled later.

Optimistic approaches have powerful advantages over pessimistic approaches. They improve availability; applications do not block when the local or remote site is down. This type of replication also permits a dynamic configuration of the network, peers can join or leave the network without affecting update propagation. There are techniques that allow such a thing as epidemic replication that propagates operations reliably to all replicas. Unlike pessimistic algorithms, optimistic algorithms scale to a large number of replicas as there is little synchronization among sites. New replicas can be added on the fly without changing the existing sites, examples are FTP and Usenet mirroring. Last but not least, optimistic approaches provide quick feedback as the system applies the updates tentatively as soon as they are submitted [36].

These advantages come at a cost with system consistency. Optimistic replication encounters the challenges of diverging replicas and conflicts between concurrent updates. For these reasons, it is used only with systems in which conflicts are rare and that tolerate inconsistent data. An example are file systems in which conflicts do not happen often due to data partitioning and access arbitration that naturally happen between users.

3.2 Data Replication Techniques in P2P

Data replication techniques in unstructured P2P networks can be classified using two criteria: techniques related with site selection and techniques related with replica distribution (see Fig. 8). Ten different replication techniques that belong to these two groups are discussed in the following.

Owner replication, Path replication and Random replication techniques are evaluated in [24].

Owner replication replicates an object only at the requesting node. When a search is successful, the object is stored at the requester node only. The number of replicas will increase in proportion to the number of requests for the service. This technique uses non-active replication and replicates the item only on the node that requested it. In this technique, the number of replicas generated in the P2P network is limited to one at each data exchange, and so it takes a large amount of time to propagate replicas over the P2P network, thereby limiting the search performance for the requested data. Owner replication is used in systems such as Gnutella.

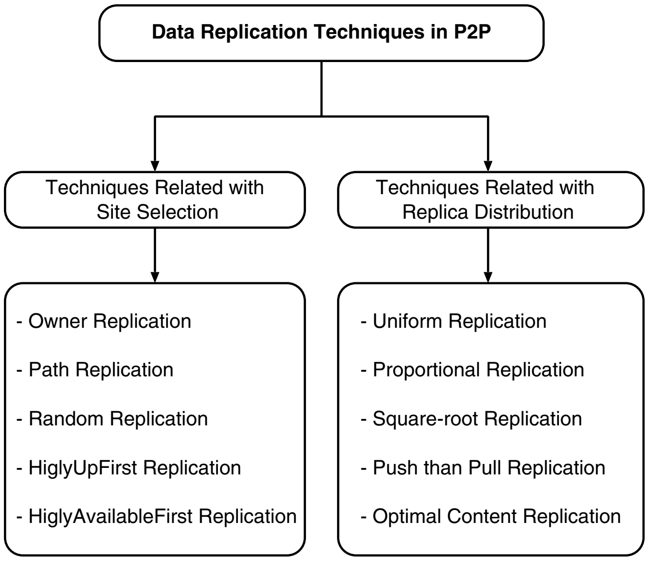

Data Replication Techniques in P2P
Techniques Related with Site Selection
Techniques Related with Replica Distribution
- Owner Replication
- Path Replication
- Random Replication
- HiglyUpFirst Replication
- HiglyAvailableFirst Replication
- Uniform Replication
- Proportional Replication
- Square-root Replication
- Push than Pull Replication
- Optimal Content Replication

Fig. 8. Data replication techniques in P2P

Path replication is a technique that uses active replication and the requested item is replicated to all nodes of the path between source and destination nodes. In this replication, the peer with a high degree forwards much more data than the peer with a low degree, so that a large number of replications will occur at the peers with a high degree. Therefore, the storage load due to writing and/or reading can be concentrated on a few high-degree peers, which thus play an important role in the P2P system. If the system fails due to overload or some other reason, a large amount of time is needed to recover the system [25]. However, this scheme has been employed in many distributed systems because of its good search performance and ease of implementation. Path replication is used in systems such as Freenet [26].

Random replication distributes the replicas in a random order. In random forwarding n-walkers random walk, random replication is the most effective approach for achieving both smaller search delays and smaller deviations in searches. Random replication is harder to implement, but the performance difference between it and path replication highlights the topological impact of path replication.

In [27], authors use developed two heuristic algorithms (HighlyUpFirst and HighlyAvailableFirst) for solving the replica placement problem and improving quality of availability.

In the **HighlyUpFirst** replication the nodes with the highest uptime are put in the set that will receive the replica.

The **HighlyAvailableFirst** method fills the so-called replica set with the nodes that have a high availability. This technique deals only with availability and does not take into consideration the network overhead.

Uniform replication strategy replicates everything equally. The purpose of this technique is to reduce search traffic. The replicas are distributed uniformly through the network. For each data object, approximately the same number of replicas are created. While this controls the overhead of replication, replicas may be found in places where peers do not access the files.

In **Proportional replication**, the number of replicas is proportional to their popularity. This replication is used for reducing search traffic. If a data item is popular, it has more chances of finding the data close to the site where query was submitted, but it is difficult to find not very popular data items.

In **Square-root replication**, the number of replicas of a file is proportional to the square-root of query distribution. This technique reduces the number of hops needed for finding an object.

Pull-Then-Push replication [28] is based on the following idea: the creation of replicas is delegated to the inquiring node, not the providing node. The scheme consists of two phases. The pull phase refers to searching for a data item. After a successful search, the inquiring node enters a push phase, whereby it transmits the data item to other nodes in the network in order to force creation of replicas. This technique increases network overhead because all neighbors get a copy of replica.

Optimal content replication [29] is an adaptive, fully distributed technique that dynamically replicates content in a near-optimal manner. This replication is used to maximize hit probabilities in P2P communities, taking intermittent connectivity explicitly into account. The optimal object replication includes a logarithmic assignment rule, which provides a closed form optimal solution to the continuous approximation of the problem. This technique does not take into consideration the past performance of nodes for selecting a suitable location for replication and this leads to resource wastage.

4 Replication Requirements and Solutions for Different Applications

Replicating objects to multiple sites has several issues such as selection of objects for replication, the granularity of replicas, and choosing appropriate site for hosting new replica [42].

By storing the data at more than one site, if a data site fails, a system can operate using replicated data, thus, increasing availability and fault tolerance. At the same time, as the data are stored at multiple sites, the request can find the data close to the site where the request originated, thus increasing the performance of the system. But the benefits of replication, of course, do not come without overheads of creating, maintaining, and updating the replicas. If the application has read-only nature, replication can greatly improve the performance. But, if the application needs to process update requests, the benefits of replication can be neutralized to some extent by the overhead of maintaining consistency among multiple replicas. If an application requires rigorous consistency and has large numbers of update transactions, replication may diminish the performance as a result of synchronization requirements. However, if the application involves read-only queries, performance can be enlarged [38].

4.1 Consistency and Limits to Replication

P2P systems can be used for a wide range of applications, including music and video sharing, wide-area file systems, archival file systems, software distribution. Two key properties of P2P applications that impact the use of replication are the size of the object that should be replicated and the time of replica delivery.

Replication should be transparent to the user, it has to achieve one logical view of the data. The fact that all users see the same data at any time is expressed in terms of consistency. Full consistency means that original data and its replicas are identical, while in partial consistency state there are differences or conflicts among original data and its replicas. One main issue is thus to achieve a satisfactory degree of consistency so that all users see the same data. The degree of consistency depends on many factors, but primarily it depends on whether the application or system can tolerate a partial consistency.

4.2 Context and Uses of Data Replication

Data replication arises in many contexts of distributed systems and applications.

Distributed Storage: One main context of replication is that of Distributed Database Management Systems (DBMS). With the emergence of large-scale distributed computing paradigms such as Cloud, Grid, P2P, Mobile Computing, etc., the data replication has become a commonplace approach, especially to ensure scalability to millions of users of such systems. In particular, data replication is used in Data Centers as part of Cloud Computing systems. In [39], authors propose and evaluate different replication methods for load balancing on distributed storages in P2P networks.

Disaster Management Scenarios: In these scenarios, ensuring anytime access to data is a must. The rescue teams need to collaborate and coordinate their actions and anytime access to data and services is fundamental to support teamwork because decision taking is time-sensitive and often urgent.

Business Applications: Data replication has attracted the attention of researchers and developers from businesses and business intelligence as a key technique to ensure business continuity, continuity-of-operations, real-time access to critical data as well as for purposes of handling big data for business analytics [40], [41]. In such context replication is seen as a choice to make data in a business environment operational.

Collaborative and Groupware Systems: One important requirement in collaborative and groupware systems is to support distributed teamwork, which often suffers from disruption. For example, supporting large user communities (e.g., from High Energy Physics community) during scientific collaborations projects. Replication is thus a means to ensure access to data anytime and thus support collaboration even in unreliable networking environments. This later feature is each time more important due to the mobility of the teamwork and use of mobile devices.

In P2P super-peer collaborative systems, peers are organized in peer-groups and collaborate together synchronously and/or asynchronously to accomplish a common project by sharing documents and data (contacts, calendar information, etc.), also known as P2P groupware systems. Such systems are attractive for several application contexts such as collaborative work in online teams in virtual campuses and small to medium corporates. These applications are especially interesting due to their low cost of deployment and maintenance compared to the rather high cost centralized groupware applications; also, they provide facilities to support opportunistic collaboration by giving full control to the users. P2P replication is particularly useful for P2P groupware systems and collaborative teamwork, by increasing the availability and access to all the information that the team manages, supporting thus a whole range of possibilities to accelerate, improve and make more productive teamwork [43].

5 A Fuzzy-Based System for Evaluating Data Replication Factor

5.1 Fuzzy Logic

Fuzzy Logic (FL) is the logic underlying modes of reasoning which are approximate rather then exact. The importance of FL derives from the fact that most modes of human reasoning and especially common sense reasoning are approximate in nature. FL uses linguistic variables to describe the control parameters. By using relatively simple linguistic expressions it is possible to describe and grasp very complex problems. A very important property of the linguistic variables is the capability of describing imprecise parameters.

The concept of a fuzzy set deals with the representation of classes whose boundaries are not determined. It uses a characteristic function, taking values usually in the interval [0, 1]. The fuzzy sets are used for representing linguistic labels. This can be viewed as expressing an uncertainty about the clear-cut meaning of the label. But the important point is that the valuation set is supposed to be common to the various linguistic labels that are involved in the given problem.

The fuzzy set theory uses the membership function to encode a preference among the possible interpretations of the corresponding label. A fuzzy set can be defined by examplification, ranking elements according to their typicality with respect to the concept underlying the fuzzy set [44].

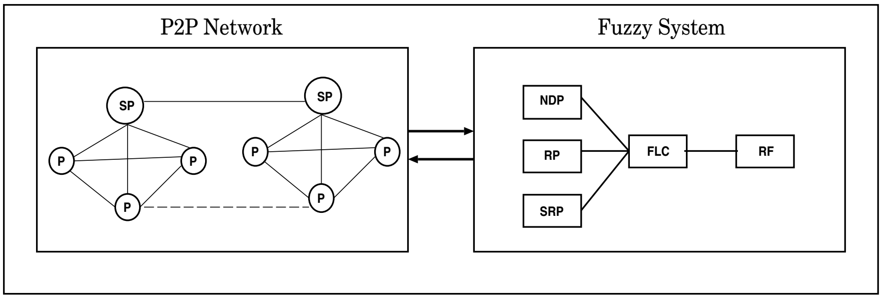


Fig. 9. Fuzzy-based system for evaluating replication factor

5.2 Proposed System

This section presents the architecture of a fuzzy-based system for evaluating replication factor in a P2P network. The structure of our system is shown in Fig. 9.

In our proposed system, we considered P2P systems with super-peer (SP) architecture. A peer-group has multiple peers that can be geographically far away from one another but have a common goal. For example, they could work together accomplishing a certain job. Job consists of several tasks that are to be completed by peers in the group. One of the most important tasks is the replication of the documents among peers of the group. Each peer can directly communicate with any other peer in the group.

In this work we considered peer-groups with the same number of peers. The peer-group has a central manager which is the super-peer. The super-peer assigns tasks to the peers in the group and keeps track of accomplishing the job. The super-peer facilitates the communication with other peers in other peer-groups. The advantage of having a super-peer is that job submissions and data queries arrive faster to the destination. The super-peer makes the connection between the peers in the group and the other peers and super peers from the network. If a part or the document in a peer changes, other peers that have the replica of this document make the changes.

During replication it is important to find a right replication factor. The replication factor is considered the total number of replicated documents over the total number of documents which means the sum of the replicas and original documents in all peers. Careful planning should be done when deciding which documents to replicate and at which peers.

Our fuzzy-based system uses three input parameters which are read from P2P network: Number of Documents per Peer *(NDP)*, Replication Percentage *(RP)*, and Scale of Replication per Peer *(SRP)*. The output parameter is Replication Factor *(RF)*.

The membership functions for our system are shown in Fig. 10. In Table 1, we show the Fuzzy Rule Base (FRB) of our proposed system, which consists of 27 rules.

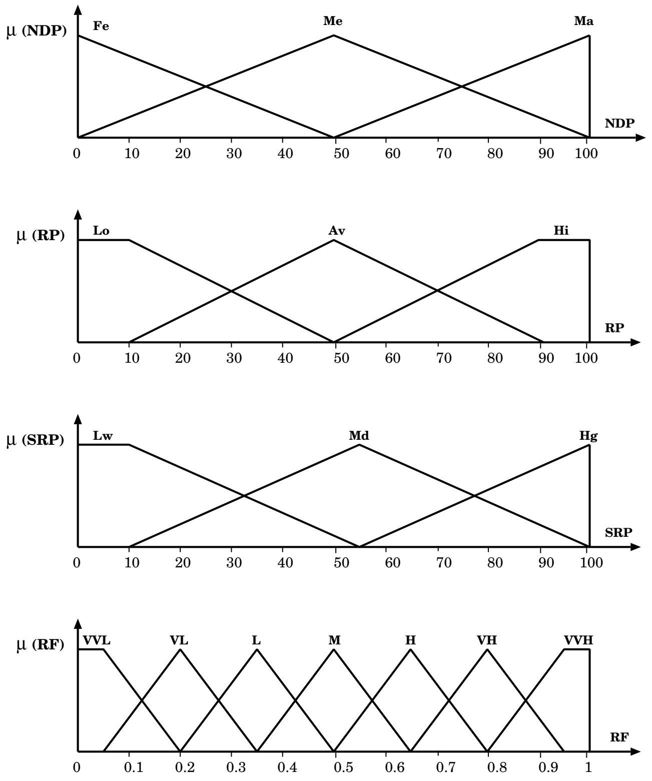


Fig. 10. Membership functions

The term sets of *NDP*, *RP*, and *SRP* are defined respectively as:

$$\begin{aligned}\mu(NDP) &= \{Few,\ Medium,\ Many\}\\ &= \{Fe,\ Me,\ Ma\};\\ \mu(RP) &= \{Low,\ Average,\ High\}\\ &= \{Lo,\ Av.\ Hi\};\\ \mu(SRP) &= \{Low,\ Medium,\ High\}\\ &= \{Lw,\ Md,\ Hg\}.\end{aligned}$$

and the term set for the output (*RF*) is defined as:

$$\begin{aligned}\mu(RF) = \{&Very\ Very\ Low,\ Very\ Low,\ Low,\ Middle,\ High,\\ &Very\ High,\ Very\ Very\ High\}\\ = \{&VVL,\ VL,\ L,\ M,\ H,\ VH,\ VVH\}.\end{aligned}$$

Table 1. FRB

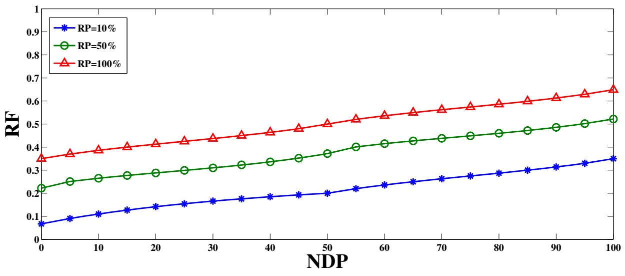
Rules	NDP	RP	SRP	RF
0	Fe	Lo	Lw	VVL
1	Fe	Lo	Md	VL
2	Fe	Lo	Hg	L
3	Fe	Av	Lw	VL
4	Fe	Av	Md	L
5	Fe	Av	Hg	M
6	Fe	Hi	Lw	L
7	Fe	Hi	Md	M
8	Fe	Hi	Hg	H
9	Me	Lo	Lw	VL
10	Me	Lo	Md	L
11	Me	Lo	Hg	M
12	Me	Av	Lw	L
13	Me	Av	Md	M
14	Me	Av	Hg	H
15	Me	Hi	Lw	M
16	Me	Hi	Md	H
17	Me	Hi	Hg	VH
18	Ma	Lo	Lw	L
19	Ma	Lo	Md	M
20	Ma	Lo	Hg	H
21	Ma	Av	Lw	M
22	Ma	Av	Md	H
23	Ma	Av	Hg	VH
24	Ma	Hi	Lw	H
25	Ma	Hi	Md	VH
26	Ma	Hi	Hg	VVH

5.3 Simulation Results

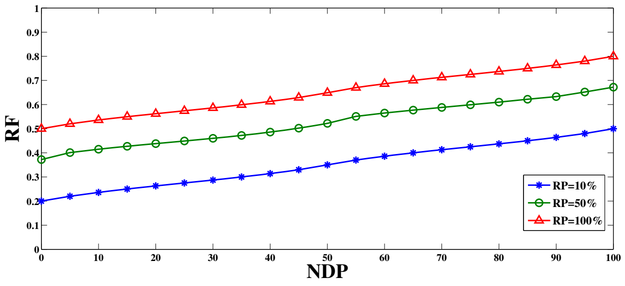
We evaluate the proposed system by simulations. The simulations are carried out using MATLAB. In Fig 11(a), we show the relation between RF and NDP, RP, SRP. In this case the SRP is considered 10%. From the figure we can see that for RP=10% with the increase of the NDP the RF increases. Also, when the RP is increased the replication factor is increased.

In Fig. 11(b) are shown the simulation results for SRP=60%. As can be seen, the replication factor increases with the increase of SRP parameter.

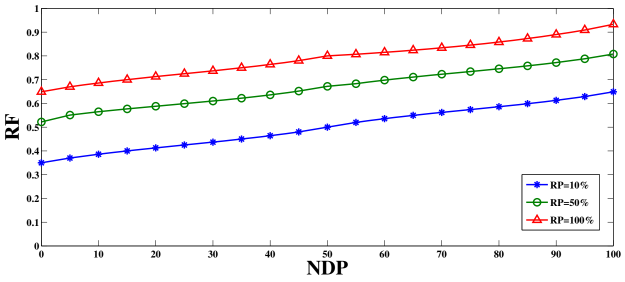
In Fig 11(c), the value of the SRP is considered 1. We can see that, with the increase of NDP and RP, the PR is increased. From the simulation results we conclude that the RF increase proportionally with increases of NDP, RP, and SRP parameters.



(a) Replication factor for SRP=10%.



(b) Replication factor for SRP=60%.



(c) Replication factor for SRP=100%.

Fig. 11. Simulation results

6 Discussion and Analysis

In P2P systems, the quality of the network will decrease if a very popular file is stored at only one available node, because of the high-network traffic rate that is caused by the amount of downloading peers. To solve this problem, replicas of popular files or objects have to be stored elsewhere in the network. If replication is implemented well, the availability, reliability, and scalability of the P2P system will increase.

6.1 Setting Up a Replication Plan

Implementing data replication requires setting up a replication plan and to answer some key questions to ensure desired properties of the system.

1. **What to replicate?** This is to identify the kind of data to replicate.
   - Full objects, fine-grained objects, chunks/blocks can be replicated.
   - Data could be documents, files, meta-data, multimedia, user profiles, events, messages, etc.
   - Data could be static or dynamic over time.
   - Replication could be meant for access purposes only or for disaster recovery as well.
   - Evaluate the homogeneity/heterogeneity degree of the data (heterogeneous vs. homogeneous data).
   - Evaluate the degree of structuring of the data (structured vs. unstructured data).

   Distributed resources should be replicated according to the resource popularities in order to maximize the probability that requests from peers will be satisfied (i.e., hit rate of requests). This is particularly true for popular resources that might have a limited number of replicas in a P2P network when first introduced, and many requests from peers can cause network congestions and slow download speed.
2. **Where to replicate?** This question has to do with the underlying computing environment where replication will take place.
   - Evaluate the heterogeneity degree of the computing environment (heterogeneous vs. homogeneous computing environment).
   - Evaluate how much storage capacity, performance, and reliability, type of storage available at replicated sites.
   - Evaluate the cost of the replication in the underlying infrastructure.

   The replicated copies should be placed in close proximity to peers who are likely to request the resource. This allows peers to be able to search and find desired resources, and reduces delays occurring during search and downloading. Also, P2P architecture is required to adapt replicas into various variations, the replication strategy should use the properties of peers and their surrounding usage environment attributes to determine which peers should be selected to perform adaptive replications and where the resulting replicas should be stored.
3. **How to replicate?** This should address the needs of:
   - How much data has to be online (to be replicated)? Will the replication be done synchronously or asynchronously?
   - How should the original data and its replicas be related? Decide the consistency type, full vs. partial replication, etc.

As in other types of large-scale distributed systems, data replication is useful to achieve important system properties due to system node failures: high availability, system reliability, and scalability. While it is well understood and easy to achieve replication of immutable information (typically files) in P2P systems, it becomes more challenging to implement data replication in techniques under highly dynamic nature of large P2P systems. Indeed, replicating documents that could change over time requires addressing the consistency issues.

Replicating objects in all sites, which significantly reduce data access cost is not realistic because it generates a large bandwidth consumption. Replicating objects to multiple sites has several issues such as: selection of objects for replication, the granularity of replicas, and choosing an appropriate site for hosting new replica. A replication scheme should manage the frequent failure of nodes in the network to provide good success rate by maintaining replicas in other suitable peers.

Data replication can also be used for maximizing hit probability of access request for the contents in P2P community, maximizing content searching (lookup) time, minimizing the number of hops visited to find the requested content, minimizing the content cost, distributing peer load.

There are a number of advantages and disadvantages to replication.

The following are the advantages of replication:

- **High availability, reliability, and fault tolerance:** Data replication means storing copies of the same data at multiple peers, thus improving availability and scalability. Full documents (or just chunks) can be replicated. Since the same data can be found at multiple peers, availability is assured in case of peer failure. Also, the throughput of the system is not affected in case of a scale-out as the operations with the same data are distributed across multiple peers.
- **Scalability:** Service capacity increased due to server load can be decreased. Response time and QoS requirements can be greatly improved.
- **Performance:** Increased performance due to data access.

The following are the disadvantages of replication:

- **Increased overhead on update:** When an update is required, a database system must ensure that all replicas are updated.
- **Require more disk space:** Storing replicas of same data at different sites consumes more disk space.
- **Expensive:** Concurrency control and recovery techniques will be more advanced and hence, more expensive. In general, replication enhances the performance of read operations and increases the availability of data to read-only transactions. However, update transactions incur greater overhead. Controlling concurrent updates by several translations to replicated data is more complex than using the centralized approach to concurrency control.

7 Conclusions

Data replication and synchronization techniques have recently attracted a lot of attention of researchers from the P2P computing community. Such techniques are fundamental to increase data availability, reliability, and robustness of P2P applications. However, several issues arise, such as data consistency and designing cost-efficient solutions, due to the highly dynamic nature of P2P systems.

This chapter conducts a theoretical survey of replication techniques in P2P systems. We describe different techniques and discuss their advantages and disadvantages. The replication techniques depends on the application in which they will be used. In general a replication technique should take into consideration at the same time: the reduction of access time and bandwidth consumption, choose an optimal number of replicas and a balanced workload between replicas. Data replication is useful to achieve high-data availability, system reliability, and scalability and can also be used for maximizing hit probability of access request for the contents in P2P community, maximizing content searching (look-up) time, minimizing the number of hops visited to find the requested content, minimizing the content cost, distributing peer load. But the benefits of replication, of course, do not come without overheads of creating, maintaining, and updating the replicas. If the application has read-only nature, replication can greatly improve the performance. But, if the application needs to process update requests, the benefits of replication can be neutralized to some extent by the overhead of maintaining consistency among multiple replicas. If an application requires rigorous consistency and has large numbers of update transactions, replication may diminish the performance as a result of synchronization requirements.

Careful planning should be done when deciding which documents to replicate and at which peers. During replication it is important to find the right replication factor.

References

1. Xhafa, F., Fernandez, R., Daradoumis, T., Barolli, L., Caballé, S.: Improvement of JXTA Protocols for Supporting Reliable Distributed Applications in P2P Systems. In: Enokido, T., Barolli, L., Takizawa, M. (eds.) NBiS 2007. LNCS, vol. 4658, pp. 345–354. Springer, Heidelberg (2007)
2. Barolli, L., Xhafa, F., Durresi, A., De Marco, G.: M3PS: A JXTA-based Multiplatform P2P System and Its Web Application Tools. International Journal of Web Information Systems 2(3/4), 187–196 (2006)
3. Arnedo, J., Matsuo, K., Barolli, L., Xhafa, F.: Secure Communication Setup for a P2P based JXTA-Overlay Platform. IEEE Transactions on Industrial Electronics 58(6), 2086–2096 (2011)
4. Barolli, L., Xhafa, F.: JXTA-Overlay: A P2P Platform for Distributed, Collaborative, and Ubiquitous Computing. IEEE Transactions on Industrial Electronics 58(6), 2163–2172 (2011)
5. Enokido, T., Aikebaier, A., Takizawa, M.: Process Allocation Algorithms for Saving Power Consumption in Peer-to-Peer Systems. IEEE Transactions on Industrial Electronics 58(6), 2097–2105 (2011)
6. Waluyo, A.B., Rahayu, W., Taniar, D., Scrinivasan, B.: A Novel Structure and Access Mechanism for Mobile Data Broadcast in Digital Ecosystems. IEEE Transactions on Industrial Electronics 58(6), 2173–2182 (2011)
7. Zhang, J., Honeyman, P.: A Replicated File System for Grid Computing. Concurrency and Computation: Practice and Experience 20(9), 1113–1130 (2008)

8. Elghirani, A., Subrata, R., Zomaya, A.Y.: Intelligent Scheduling and Replication: a Synergistic Approach. Concurrency and Computation: Practice and Experience 21(3), 357–376 (2009)
9. Nicholson, C., Cameron, D.G., Doyle, A.T., Millar, A.P., Stockinger, K.: Dynamic Data Replication in LCG. Concurrency and Computation: Practice and Experience 20(11), 1259–1271 (2008)
10. Shirkey, C.: What is P2P..and What isn't. O'Reilly Network (November 2000)
11. Gnutella, `http://gnutella.wego.com/`
12. NAPSTER, `http://www.napster.com/`
13. WinMX, `http://www.frontcode.com/`
14. FREENET, `http://frenet.sourceforge.net/`
15. GROOVE, `http://www.groove.net/`
16. Martins, V., Pacitti, E., Valduriez, P.: Survey of Data Replication in P2P Systems. Technical Report (2006)
17. Bernstein, P., Goodman, N.: The Failure and Recovery Problem for Replicated Databases. In: Proc. of the Second Annual ACM Symposium on Principles of Distributed Computing, pp. 114–122. ACM Press, New York (1983)
18. Mustafa, M., Nathrah, B., Suzuri, M., Osman, M.: Improving Data Availability Using Hybrid Replication Technique in Peer-to-Peer Environments. In: Proc. of 18th International Conference on Advanced Information Networking and Applications (AINA-2004), pp. 593–598. IEEE CS Press (2004)
19. Loukopoulos, T., Ahmad, I.: Static and Adaptive Data Replication Algorithms for Fast Information Access in Large Distributed Systems. In: Proc. of 20th International Conference on Distributed Computing Systems (ICDCS 2000), pp. 385–392. IEEE CS Press (2000)
20. Xhafa, F., Potlog, A., Spaho, E., Pop, F., Cristea, V., Barolli, L.: Evaluation of Intragroup Optimistic Data Replication in P2P Groupware Systems. Concurrency Computat.: Pract. Exper (2012), doi:10.1002/cpe.2836
21. Potlog, A.D., Xhafa, F., Pop, F., Cristea, V.: Evaluation of Optimistic Replication Techniques for Dynamic Files in P2P Systems. In: Proc. of Sixth International Conference on on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC 2011), Barcelona, Spain, pp. 259–165 (2011)
22. Xhafa, F., Kolici, V., Potlog, A.D., Spaho, E., Barolli, L., Takizawa, M.: Data Replication in P2P Collaborative Systems. In: Proc. of Seventh International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC 2012), Victoria, Canada, pp. 49–57 (2012)
23. Coulon, C., Pacitti, E., Valduriez, P.: Consistency Management for Partial Replication in a High Performance Database Cluster. In: Proc. of the 11th International Conference on Parallel and Distributed Systems (ICPADS 2005), pp. 809–815 (2005)
24. Lv, Q., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and Replication in Unstructured Peer-to-Peer Networks. In: Proc. of 16th ACM International Conference on Supercomputing (ICS 2002), pp. 84–95 (2002)
25. Keyani, P., Larson, B., Senthil, M.: Peer Pressure: Distributed Recovery from Attacks in Peer-to-Peer Systems. In: Gregori, E., Cherkasova, L., Cugola, G., Panzieri, F., Picco, G.P. (eds.) NETWORKING 2002. LNCS, vol. 2376, pp. 306–320. Springer, Heidelberg (2002)
26. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. In: Federrath, H. (ed.) Anonymity 2000. LNCS, vol. 2009, pp. 46–66. Springer, Heidelberg (2001)

27. On, G., Schmitt, J., Steinmetz, R.: The Effectiveness of Realistic Replication Strategies on Quality of Availability for Peer-to-Peer Systems. In: Proc. of the Third International IEEE Conference pn Peer-to-Peer Computing, pp. 57–64 (2003)
28. Leontiadis, E., Dimakopoulos, V.V., Pitoura, E.: Creating and Maintaining Replicas in Unstructured Peer-to-Peer Systems. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1015–1025. Springer, Heidelberg (2006)
29. Kangasharju, J., Ross, K.W., Turner, D.A.: Optimal Content Replication in P2P Communities. Manuscript, pp. 1–26 (2002)
30. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The Dangers of Replication and a Solution. In: Proc. of International Conference on Management of Data (SIGMOD 1996), pp. 173–182 (1996)
31. Lubinski, A., Heuer, A.: Configured Replication for Mobile Applications. In: Databases and Information Systems, pp. 101–112. Kluwer Academic Publishers, Dordrecht (2000)
32. Rohm, U., Bohm, K., Schek, H., Schuldt, H.: FAS - A Freshness-Sensitive Coordination Middleware for a Cluster of OLAP Components. In: Proc. of 28th International Conference on Very Large Data Bases (VLDB 2002), pp. 754–765 (2002)
33. Bernstein, P.A., Hadzilacos, V., Goodman, N.: Concurrency Control and Recovery in Database Systems (1987)
34. Kemme, B., Alonso, G.: A New Approach to Developing and Implementing Eager Database Replication Protocols. ACM Transactions on Database Systems 25(3), 333–379 (2000)
35. Pacitti, E., Minet, P., Simon, E.: Fast Algorithms for Maintaining Replica Consistency in Lazy Master Replicated Databases. In: Proc. of the 25th International Conference on Very Large Data Bases (VLDB 1999), pp. 126–137 (1999)
36. Saito, Y., Shapiro, M.: Optimistic Replication. ACM Comput. Surv. 37(1), 42–81 (2005)
37. Chundi, P., Rosenkranz, D.: Deferred Updates and Data Placement in Distributed Databases (1996)
38. Goel, S., Buyya, R.: Data Replication Strategies in Wide Area Distributed Systems. In: Enterprise Service Computing: From Concept to Deployment, pp. 211–241. IGI Global (2007)
39. Yamamoto, H., Maruta, D., Oie, Y.: Replication Methods for Load Balancing on Distributed Storages in P2P Networks. The Institute of Electronics, Information and Communication Engineers E-89-D(1), 171–180 (2006)
40. Sheppard, E.: Continuous Replication for Business-Critical Applications. White Paper, pp. 1–7 (2012)
41. Van Der Lans, R.F.: Data Replication for Enabling Operational BI., White Paper on Business Value and Architecture, pp. 1–26 (2012)
42. Ulusoy, O.: Research Issues in Peer-to-Peer Data Management. In: Proc. of International Symposium on Computer and Information Sciences (ISCIS 2007), pp. 1–8 (2007)
43. Estepa, A.N., Xhafa, F., Caballé, S.: A P2P Replication-Aware Approach for Content Distribution in e-Learning Systems. In: Proc. of Sixth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2012), pp. 917–922 (2012)
44. Terano, T., Asai, K., Sugeno, M.: Fuzzy Systems Theory And Its Applications. Academic Press, Inc., Harcourt Brace Jovanovich Publishers (1992)