

BABEȘ-BOLYAI UNIVERSITY
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Huge Distributed File Systems On Unix Platforms

Abstract

PhD student: Dan Cojocar

Scientific supervisor: Prof. Dr. Florian Mircea Boian

Cluj-Napoca

2015

List of publications

- [1] Florian Mircea Boian, Darius Vasile Bufnea, **Dan Cojocar**, Alexandru Ioan Vancea, and Adrian Sterca. “A Model for Efficient Session Object Management in Web Applications”. In: *Proceedings of the Symposium Colocviul Academic Clujean de Informatica*. Cluj–Napoca, Romania, June 2006, pp. 137–142.
- [2] **Dan Cojocar**. “Hardware Fault-Tolerant File System”. In: *Proceedings of the Symposium Colocviul Academic Clujean de Informatica*. Cluj–Napoca, Romania, June 2006.
- [3] Ioan Lazar and **Dan Cojocar**. “On Model-Driven Development for web Applications”. In: *Studia Universitatis Babeş-Bolyai, Informatica* (2006), pp. 101–112.
- [4] Boian Florian Mircea, Darius Vasile Bufnea, Claudiu Cobarzan, Alexandru Ioan Vancea, Adrian Sterca, and **Dan Cojocar**. *Sisteme de operare*. RISOPRINT, 2006.
- [5] Florian Mircea Boian, Darius Vasile Bufnea, Alexandru Ioan Vancea, Adrian Sterca, **Dan Cojocar**, and Rares Florin Boian. “Some Formal Approaches for Dynamic Life Session Management”. In: *Proceedings of Knowledge Engineering: Principles and Techniques Conference (KEPT)*. Cluj–Napoca, Romania, June 2007, pp. 227–235.
- [6] **Dan Cojocar**. “BBUFs: A new lookup mechanism based on IPV6”. In: *Workshosp on Global Computing Models and Technologies, co-located with SYNASC 2008*. Timisoara, Romania, 2008, pp. 358–361. (**ISI Proceedings**).
- [7] **Dan Cojocar**. “BBUFs: Synchronization Mechanism”. In: *6th International Conference of Applied Mathematics (ICAM)*. Baia Mare, Romania, 2008, pp. 363–368.
- [8] Rares Florin Boian and **Dan Cojocar**. “Moving Excess Data Into External Peer-to-Peer Storage”. In: *KEPT2009 Knowledge Engineering Principles and Techniques Selected Papers* (2009), pp. 358–365. (**ISI Proceedings**).
- [9] Rares Florin Boian and **Dan Cojocar**. “Moving Excess Data Into External Peer-to-Peer Storage”. In: *Proceedings of Knowledge Engineering: Principles and Techniques Conference (KEPT)*. Cluj–Napoca, Romania, July 2009, pp. 296–299.

- [10] **Dan Cojocar**. “BBUFs: Architecture Overview”. In: *Proceedings of Knowledge Engineering: Principles and Techniques Conference (KEPT)*. Cluj–Napoca, Romania, July 2009, pp. 276–279.
- [11] **Dan Cojocar**. “The Architecture of BBUFs”. In: *KEPT2009 Knowledge Engineering Principles and Techniques Selected Papers (2009)*, pp. 335–342. (**ISI Proceedings**).
- [12] **Dan Cojocar** and Florian Mircea Boian. “BBUFs: Replication Strategies”. In: *Proceedings of Knowledge Engineering: Principles and Techniques Conference (KEPT)*. Cluj–Napoca, Romania, July 2009, pp. 284–287.
- [13] **Dan Cojocar** and Florian Mircea Boian. “BBUFs: Replication Strategies Comparison”. In: *KEPT2009 Knowledge Engineering Principles and Techniques Selected Papers (2009)*, pp. 343–350. (**ISI Proceedings**).
- [14] **Dan Cojocar**. “BBUFs: Routing Protocol”. In: *Proceedings of the Symposium Colocviul Academic Clujean de Informatica*. Cluj–Napoca, Romania, June 2010, pp. 116–121.
- [15] Grigoreta Sofia Cojocar and **Dan Cojocar**. “A Comparison of AOP Based Monitoring Tools”. In: *Studia Universitatis Babeş-Bolyai, Informatica* (2011), pp. 65–70.
- [16] **Dan Cojocar**. “Replication Location Decisions”. In: *Nano, Information Technology and Reliability (NASNIT), 2011 15th North-East Asia Symposium*. Macao, China, Oct. 2011, pp. 161–165. (**indexed IEEE**).
- [17] **Dan Cojocar**. “Load Balance Queries in Decentralized Peer-to-Peer File Systems”. In: *Proceedings of the National Symposium ZAC2012 (Zilele Academice Clujene, 2012)*. 2012, pp. 105–110.
- [18] Gabriel Ciobanu and **Dan Cojocar**. “Expressing BBUFs lookup using the π -calculus”. In: *Workshosp on Global Computing Models and Technologies, co-located with SYNASC 2014*. Timisoara, Romania, 2014. (**ISI Proceedings**).

Contents

Introduction	5
1 Distributed File Systems	7
1.1 Peer-to-Peer Systems	7
1.2 Models to Represent Decentralized Systems	8
1.2.1 Distributed Hash Tables	8
1.2.2 Systems that are using DHT	8
1.3 Distributed Peer-to-Peer File Systems	9
1.4 Lookup problem	9
1.5 Query Load Balancing	10
1.5.1 Related Approaches	10
1.6 Replication	10
1.7 Existing Decentralized Distributed File Systems	11
2 A new Approach for Decentralized Unstructured Peer-to-Peer Systems	12
2.1 IPv6 Anycast Addresses	12
2.2 Proposal	13
2.2.1 The Lookup Operation	16
2.2.2 Communication Protocol	17
2.2.3 Advantages/Disadvantages	17
2.3 Balansarea cererilor	18
2.3.1 MCP - Maximum Computing Power	18

2.3.2	AQK - Average Query Cost	19
2.4	BBUFs Case Study	20
2.4.1	System Architecture	20
2.4.2	BBUFsMapper	21
2.4.3	BBUFs Replication	21
2.5	Conclusions and Further Works	22
3	New Replication Strategies	23
3.1	Location Aware Replication	23
4	Formal Model	25
4.1	Π -calculus	25
4.1.1	Lookup Details	28
4.1.2	Timed π -calculus	29
4.1.3	Model Validation	30
4.2	Mobile Ambients	30
4.2.1	Base Concepts	31
4.2.2	Domain Mobile Ambients Behavior	31
4.2.3	Extending Domain Attribute of an Mobile Ambient	32
	Conclusions	34

Introduction

This paper represents the scientific results under the operating system domain, in special under the distributed decentralized unstructured file systems. The research started in 2005 under the supervision of Prof. Dr. Florian Mircea Boian.

In this paper we are focusing on presenting the problems that arise on lookup operations in such decentralized systems. Specifically we are trying to study the problems that are related with searching the content in such systems. We are studying how to enhance the system performance using replication and query load balancing techniques using the created replicas. Also we are presenting a proposal of a new formalism, used to model the interactions in such a system. The paper is structured in 4 chapters as follows.

The Chapter 1, **Distributed File Systems**, presents general facts about decentralized distributed peer-to-peer file systems, lookup generic problems in such systems, the causes that are produced by such issues and how to address them.

The Chapter 2, **A new Approach for Decentralized Unstructured Peer-to-Peer Systems**, presents a new original approach on the lookup problem, based on IPv6 addressing scheme. We are presenting the novelty introduced by the IPv6, how the anycast and multicast addressing schemes were redefined. And then how we are using them in our proposal. Also we are presenting the architecture of a decentralized unstructured file system, named BBUFs, that is using our proposal. In this chapter we are presenting a new way to load balance client queries using multiple replicas.

The Chapter 3, **New Replication Strategies** introduces a new replication method that allows us to select or specify the criteria used to select a replica destination.

The Chapter 4, **Formal Model**, presents a formal model for the lookup protocol that we defined. We are presenting also some related approaches and a new enhancement to the *Time and Space Coordination of Mobile Agents* formalism. Also we are presenting the formal model presented in Chapter 2 and a way to validate the model.

The original contributions that were used in this paper are found in the following chapters: 2, 3, 4, and are proposing:

- A new approach to the lookup problem in a decentralized unstructured file system [Coj09, Coj08a] (Section 2.2).
- A design for a new decentralized distributed file system, based on the lookup approach [Coj09, Coj08b] (Section 2.4)
- A new way to load balance the client queries using multiple replicas [Coj12, Coj10] (Section 2.3.1).
- A new model to represent the internal structure that will allow the distribution of the file content among multiple nodes [Coj08b] (Section 2.4.3).
- A new replication strategy that will take into account the nodes locations, using different metrics [CB09a, Coj11] (Section 3.1).
- A formal proposal of the lookup mechanism using *pi*-calculus [CC14] (Section 4.1.1).
- A new formalism by extending the domain attribute of the mobile ambients defined in *Time and Space Coordination of Mobile Agents* (Section 4.2.3).

Chapter 1

Distributed File Systems

1.1 Peer-to-Peer Systems

Almost unknown ten years ago, peer-to-peer systems have started to be a big Internet traffic source.

Lv [LCC⁺02a] classified peer-to-peer systems, based on their architecture, in the following categories:

- *centralized* - systems that have a central node that manages the access trout the system.
- *decentralized* - in this category none of the nodes have the absolute control.

This category has two subcategories:

- *structured* - between the system nodes and the data that they are managing there is a tight coupling.
- *unstructured* - systems that are not structured based on the data that is managed.

We are not trying to highlight any features but we are not able not to notice that the *unstructured* systems are more popular then among the others. Especially of their decentralized nature these systems are able to offer quality of service to their customers, having: smaller response times and higher transfer rates [Ora01].

1.2 Models to Represent Decentralized Systems

Most of the decentralized implementations are using distributed hash tables to represent the nodes by adding another layer on top of the network one. The added layer is used to resolve the lookup operations and to organize the information in the system. But a big disadvantage of this approach is that each time a node is entering the system or an existing one is leaving, the layer has to be updated. Next are present, the base concepts that are involved when using a distributed network layer.

1.2.1 Distributed Hash Tables

The concept of distributed hash tables (DHT) was introduced by Litwin, Niemat and Shneider in 1996 [LNS96]. Starting with 2001 the first drafts of distributed hash tables that are using huge networks have emerged.

A distributed hash table is like a huge hash map that is split among multiple interconnected computers that are scattered all over the glob. The computers, named nodes, are forming the system network. The object set that is maintained in the DHT is disjointly partitioned, each node being responsible of a single such partition. The set of nodes that are involved in a DHT network has a dynamic behavior: at any moment in time a new node can join the set or an existing one can leave.

1.2.2 Systems that are using DHT

The first systems that are using DHT started to appear in 2001 and are based on results published in 1997:

- Consistent hashing - a special hash type, that ensures that when the hash table is modified only K/n keys need to be involved. K being the key number and n the number of free positions in the hash table.
- Plaxton Mesh scheme (or PRR) - is a concept that allows an efficient routing at the node level, using small routing tables [PRR97].

The approaches used to build a distributed hash table are, usually, differentiating among them selves by the way that they are associating the ID to the node and on how the DTH is associated with each node.

1.3 Distributed Peer-to-Peer File Systems

Peer-to-Peer distributed file systems are unstructured decentralized systems [Ber03], where the nodes are hosts that use commodity hardware [LM09]. All the nodes of the systems have the same capabilities and the same duties, the communications being direct between nodes. The major features, that needs to be considered when implementing a distributed decentralized file system, are:

- Data lookup.
- A plan to ensure access to data even if the systems nodes are having issues, like hardware failures, etc.
- Efficient resource management.
- How to secure the user content.

1.4 Lookup problem

A major problem in a distributed peer-to-peer decentralized system is how to lookup user content. How can we pinpoint the user, without having a registry or a central node, to the proper nodes that are maintaining the his data? [BKK⁺03]

A simple mode to describe the lookup problem is: A client uploads a file into the system, or a node is publishing a content; At a later moment is time another client is willing to use that file. How is this new client able to locate the node that was used to store the initial file or one of his replicas?

1.5 Query Load Balancing

1.5.1 Related Approaches

Using different replication strategies (presented in Section [LM09]), most of the systems are using a big number of replicas, mainly to offer an improved performance, but also as a backup option in case of hardware failures [WJH97, LSSD02, HSW94].

The existing approaches can be classified in the following categories:

- *Static* - The system decides where to perform the operation, on each read or write call.
- *Dynamic* - The systems are designed to handle a big number of user queries. Using a static approach being difficult to handle such cases, especially if the system needs to evaluate each call.

The dynamic approaches are mainly used by the structured systems [CDG⁺08, SKRC10] since in this case they are able to take advantage of the structured nature of the system. The central node is the node that will load balance the user queries. When a client is making a new query, the central node will guide him to the best node that can provide an answer. Because of this all the improvements and decisions are done at the central node [HGM10].

1.6 Replication

Peer-to-peer unstructured and decentralized systems and applications are distributed systems that are not having a central node or a structure to manage the activities among the system. Also being unstructured all the decisions are done at the node level [LCC⁺02b]. Because of this all the nodes in the system are having the same level of decision.

Also, the nodes being transient (may appear or disappear from the system at any time), the systems have to maintain multiple copies of the same content on different nodes [PC04].

Even when maintaining a single copy is creating another type of problems: *data consistency*. When the status of a content is altered, by updating or appending operations, all the replicas need to be notified accordingly. To resolve such issues a new mechanism, named *data synchronization*, was introduced [MK03].

1.7 Existing Decentralized Distributed File Systems

There are multiple attempts to build a distributed file system. Among them, systems like: Netware [MMP94], NFS [SGK⁺85], Andrew [HKM⁺88] and Sprite [NWO88] that use a central node, that is responsible to serve all the client queries. Models that in order to ensure the good behavior of the central node is forcing performance and security constraints. To avoid such disadvantages the peer-to-peer systems are following the decentralized approach. Among such system we have: OceanStore [KBC⁺00], Ivy [MMGC02], Freenet [CSWH01], PAST [RD01] or CFS [DKK⁺01]. Each of the having their advantages and disadvantages.

Chapter 2

A new Approach for Decentralized Unstructured Peer-to-Peer Systems

In this chapter we are proposing a new approach for an peer-to-peer decentralized unstructured system that will relay on IPv6 [DH98]. In this system the nodes are not maintaining any network structure or informations about the other nodes from the system. In the next section we are presenting the IPv6 anycast concept, the new approach that we are proposing and then we are presenting Babeş Bolyai University Filesystem (BBUFs) as a case study.

2.1 IPv6 Anycast Addresses

Internet Protocol Version 6 (IPv6) [DH98] was developed to solve the problem of address space exhaustion [Tra95] on the Internet Protocol Version 4 (IPv4) [Pos81]. IPv6 specification defines a new addressing scheme called "anycast address" that is an identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is routed to the "nearest" interface having that address, depending on the distance of the routing path.

Chi-Yuan Chang et al. [CCC⁺05] have also used the anycast characteristics of IPv6 and designed an active router using fuzzy logic to receive the load information. The information can be used to determine the server mechanism used to solve the

load balance problem.

2.2 Proposal

In this section our approach on building a decentralized unstructured file system based on IPv6 anycast addresses.

BBUFs differentiates itself from these file systems because is not using a structure like Chord or DHash to build or store the overlay network of the file system. We are using a new mechanism based on IPV6 addressing scheme. Also, each node has no knowledge about other nodes except how to contact a node from his group.

In our approach a group is a set of nodes that are replicating the same content (see Figure fig:multicast). This means that a node can be part of many such groups. The group is represented by:

- an IPV6 address that will be allocated from the subnetwork dedicated to anycast use;
- a multicast IPV6 address that will be allocated from the multicast subnetwork corresponding to the anycast address defined above.

The BBUFs system is using three types of addresses:

- *unicast* - used for data communication between clients and nodes of the file system;
- *anycast* - used for lookup messages between clients and system nodes;
- *multicast* - used between the system nodes (i.e., when synchronizing their shared content).

In our system each node will have at least one address from each above defined group (see Figure 2-2):

- *an unicast address* - representing this node on our file system network. This address is assigned by the file system administrator.

- *at least one anycast address* - representing at least one directory shared by this node. This address is computed automatically using the BBUFsMapper algorithm, defined in Subsection 2.4.2.
- *at least one multicast address* - representing the multicast group that the current node is registered on. This address is also computed by the BBUFsMapper algorithm.

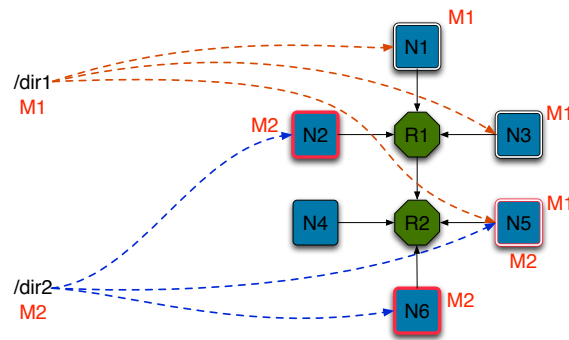


Figure 2-1: Each node is part of a group that represents the shared content.

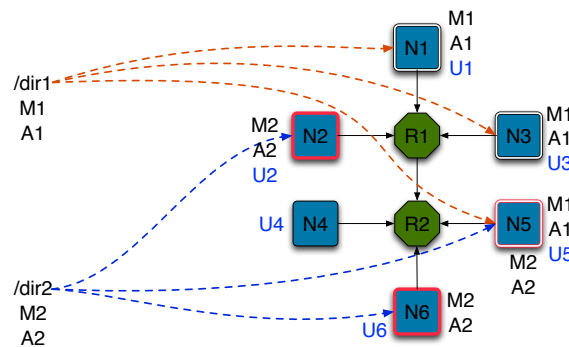


Figure 2-2: The addresses that a node is using.

Using the IPV6 addressing scheme approach we eliminate the need of building and maintaining an overlay network structure found in all peer-to-peer implementations based on Chord or DHash. We also optimize the lookup mechanism [Coj08a]. As a consequence of using IPV6 addressing scheme we gain the following benefits:

- there is no need to duplicate the routing logic;

- when a new node joins the system we do not have to create data migration logic (like in DHash [DKK⁺01]);
- the addressing space of IPV6 is considerably larger than IPV4 based implementations;
- using anycast and multicast, the lookup is performed only on a restricted set of nodes.
- using replication operations the group members are grouping, but using a single multicast request we can contact all the members [Coj08b].

Our idea is to narrow the search area in order to optimize the lookup times. To restrict the search area we use IPV6 addressing scheme and a new manner to categorize the shared information. In order to categorize the information we propose a method that binds the directory name to an IPV6 address. The directory name is transformed into an IPV6 address using a hash function (eg. SHA1 [EJ01] and BASE64 [Jos03]) on the shared content and a prefix assigned to distinguish the file system. We convert the directory names into anycast and multicast addresses in order to obtain the group of nodes that share the same content. Because of the IPV6 representation, the number of bits that we have available to represent a group is limited to 128 minus the mandatory multicast prefix, which is 16 bits long, resulting in 112 bits to represent the prefix of the network and the value of the hash function for a given directory (see Figure 2-3).

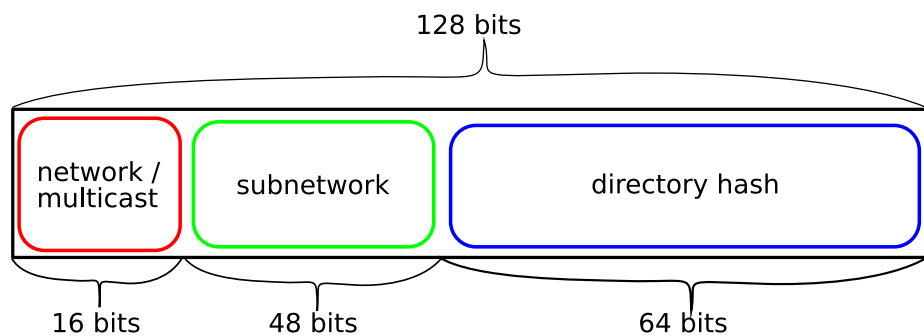


Figure 2-3: Representing a directory name using an IPv6 address.

That is why we propose that the prefix should be at most 48 bits long, that will give us 2^{48} different networks, and 64 bits to represent the hash result.

In order to prevent generating a lots of IPV6 addresses for a host we limit the depth of the lookup path of the shared content to a predefined number of directories. The value of the depth will be established by the administrator of the file system, and it will be known by all the participants (file system nodes and clients). The node administrator and the clients can increase the value of the depth to more precisely categorize the shared content.

2.2.1 The Lookup Operation

In the following we present a small example on how the lookup is performed. Suppose we have a file system represented like in Figure 2-4, where Nx are nodes ($x \in \{1..3\}$), Ry are the routers ($y \in \{1..2\}$) and C is the client. The client C needs to perform a lookup for file $/d1/d2/f2$.

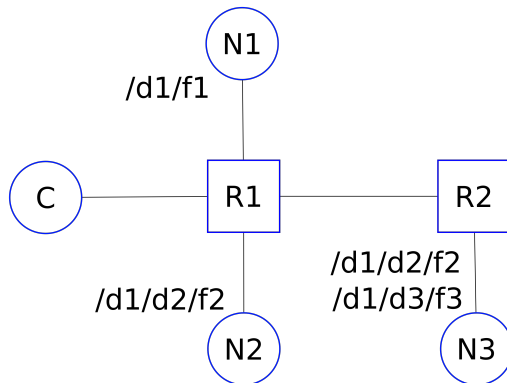


Figure 2-4: File System sample.

Using the BBUFsMapper algorithm and the predefined system variables: network prefix, depth and hash function it will determine the anycast address of a potential node that stores the searched file.

2.2.2 Communication Protocol

In this subsection we present some of the most important communication details in the BBUFs file system. We describe the steps performed for some of the most important use cases: node joining, node leaving and client lookup.

2.2.3 Advantages/Disadvantages

In this subsection we present the advantages and disadvantages of our IPv6 based lookup mechanism. The approach that we have proposed has the following advantages:

- We do not have to build an overlay network.
- The nodes from the system do not retain links or information about other nodes.
- In order to determine the node containing the data, the number of nodes reached is significantly lower (only the nodes from the determined group are contacted in worst case scenario).

The disadvantages of our approach are:

- The system depends on IPV6 capable unix machines. However, nowadays all major unix distributions are IPV6 capable.
- A node may have many IPV6 addresses resulting in big routing tables. This can be avoided using lower values for depth in BBUFsMapper algorithm (Subsection 2.4.2).
- When hash collisions appear, the system generates multicast messages that introduce latency. This can be avoided if we lower the number of bits allocated to the prefix part and we increase the number of bits allocated to the hash part. Also when a node joins and will detect that the content is different the system will initiate a merge routing, resulting in a content synchronization.

2.3 Balansarea cererilor

În această secțiune descriem cum se realizează balansarea cererilor pentru abordarea noastră de sisteme de fișiere peer-to-peer descentralizate nestructurate. Propunerea noastră încearcă să aducă avantajele enumerate în Secțiunea 1.5.1 și la sistemele peer-to-peer descentralizate nestructurate. Pentru a ne asigura că noua abordare aduce aceste avantaje, trebuie să stabilim următoarele:

- Unde se stabilesc deciziile de balansare în cazul operațiilor de depunere sau modificare de conținut?
- Dacă la sistemele centralizate nodul central menține și folosește registrele care ajută la luarea deciziei, unde vor fi gestionate aceste registre în cadrul unui sistem nestructurat descentralizat?

2.3.1 MCP - Maximum Computing Power

The results that Dong Xuan and his team presented that using anycast addresses in a routing algorithm is a viable solution in managing the requests for nodes that are using such addressing schemes [XJZZ00].

Using these results, Han Zhi-nan defined two metrics to be used by such a routing algorithm, to establish an optim path [ZnWLY11].

Similarly, we defined a new metric in [Coj10], named *Maximum Computing Power (MCP)*. The metric is taking into account the query type and how the system was behaving while processing the this type of queries.

We are defining the MCP for a node n as follows: $MCP(n) = ri - k * no_cores$, where:

- n - Represents the node for which the value is computed.
- ri - Represent the number of running instances of *BBUFsRepository* module.
- k - Is a constant that the node system administrator will be able to define.
- no_cores - Is the number of cpu cores that the node has.

Using the same processing logic detailed in [ZnWLY11] we will be able to redirect the client request to the “nearest” idle node and control the loading of our nodes.

2.3.2 AQK - Average Query Cost

In [Coj12] using the MCP results and the informations that are collected by the system, we are presenting how each node is able to compute an average query cost, for each node.

Definition 1 The cost of operation o is defined like:

$$operationType(o) * contentSize$$

where:

- *Operation cost (OK)* - will represent the cost of the operation. Probably the write operation will be more expensive than a read one, depending on the underlying storage system capabilities.
- *Content Size (CS)* - the size of the query payload.

By computing and associating, such a cost for each incoming request, with the actual time spend (AST) for handling the query, the system will be able to establish a cost for each query (QK) and average query cost (AQK) per query type, like:

Definition 2 Average query cost AQK is defined:

$$AQK = \frac{\sum_{i=1}^N QK(i)}{\sum_{i=1}^N CS(i)}$$

where: $QK(i) = OK(i) * CS(i) * AST(i)$ represent the cost of i served query.

By using the MCP values a router is able to establish what are the "idle" nodes. Using the query type and choosing the node with the lowest AQK costs for that type, the routing algorithm will be able to select the best "idle" node that will resolve the query.

2.4 BBUFs Case Study

BBUFs (Babeş Bolyai University File System) is a peer-to-peer decentralized file system that has similar characteristics to Ivy [MMGC02] and Pastis [BPS05]:

- *highly scalable* - benefit of decentralization of peer-to-peer system;
- *fault tolerant* - a file can be configured to have n number of replicas on different nodes;
- *highly performable* - each client "talks" directly to the nearest node that is providing the requested content.

2.4.1 System Architecture

Using the information from *BBUMeta* presented in Figure 2-5 [Coj08b], the fundamental object in BBUFs system, we are able to determine how many copies each shared content needs our distributed system to maintain.

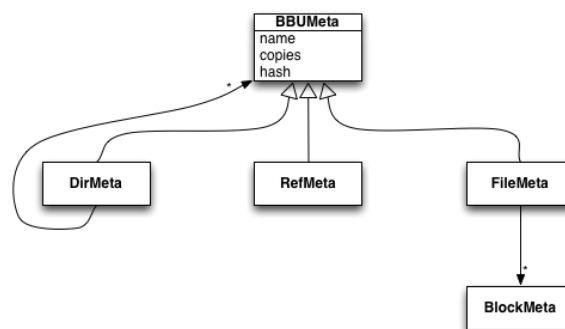


Figure 2-5: Metadata classes.

Another key component of the system is *BBUFsRepository* (see Figure 2-6). This component is running on each node and it is responsible of serving the client queries [CB09b].

Using the components that were presented until now we can try to answer the following question: *What is happening when we have a corrupted copy?*

If one of our system goals is to be able to survive even to situations where a whole subnetwork is not accessible, we need to focus also on where are we replicating

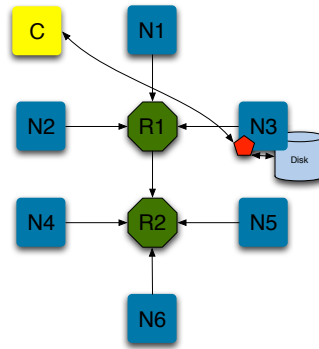


Figure 2-6: BBURepository - Serves client requests.

the content, not just on how to improve the client query response times and how many replicas to create. In Chapter 3 we present our proposal on how a peer-to-peer *decentralized unstructured* system can achieve this goal.

2.4.2 BBUFsMapper

In this subsection we describe the BBUFsMapper algorithm that is used to bind the directory name to an IPV6 address.

2.4.3 BBUFs Replication

BBUFs file system is using multiple replicas to avoid data losses and to maintain a copy as close as possible to the clients. This way ensuring that the clients will benefit from higher transfer rates [CB09b, Coj11].

In our approach a group is a set of nodes that are replication the same content.

In this approach we are also trying to answer the following questions:

- How can we synchronize a directory?
- How to establish the replica destination?
- How to detect if a copy has modified her content?
- On what node do we have the latest version?

Algorithm 1 BBUFsMapper algorithm

Input:

- *type* - address type to be generated (anycast or multicast)
- *dir* - the full directory name
- *depth* - the predefined directory depth
- *prefix* - the network prefix used for this file system
- *hash* - the hash function

Output

- *address* the generated address

Algorithm *BBUFsMapper* is:**Begin****if** $computeDepth(dir) > depth$ **then** $dir \leftarrow determineParentDir(dir, depth)$ \triangleright extract the parent directory of *dir* that has *depth* length.**end if** $key \leftarrow hash(dir)$ \triangleright return the hash value of the string *dir***if** *type* is anycast **then** $address \leftarrow "FEC0:" + prefix + key$ **else** $address \leftarrow "FF08:" + prefix + key$ **end if****End.**

2.5 Conclusions and Further Works

In this chapter we presented a new approach in designing a decentralized unstructured distributed file system. A system that is using the new IPv6 lookup mechanism that we introduced in the previous chapter. We presented a part of the most important operations that the system needs: the steps needed when a node is joining the system, what is happening when a node is leaving the system, etc. Also we presented the advantages and the disadvantages of our approach.

Chapter 3

New Replication Strategies

In this chapter we are presenting a new approach that is solving the problems presented in Section 1.6, using the BBUFs [Coj09] as a case study. BBUFs is a peer-to-peer system that is using IPv6 [DH98] as a network layer to solve the persisting framework. By design, the BBUFs client queries are resolved by the right node, since the lookup mechanism has $O(1)$ complexity in most of the cases [Coj08a]. Section 3.1 presents a new replication strategy to identify the best location for a new replica. Some conclusions and further works, related to this topic, are presented.

3.1 Location Aware Replication

Using the *BBUFsStatistics* module (see Figure 3-1), a process that is collecting data about the system behavior, the system is able to establish when we need a new replica [CB09b].

Using *SyncDaemon* and *BBUFsStatistics* the system is able to trigger a new replication process, based on the following strategies:

- *Replication based on access counters* - once a threshold values is surpassed it will start the process.
- *Replication based on weights* - the system is taking into account the parent-child hierarchy of the shared files and once a parent needs to be replicated it may

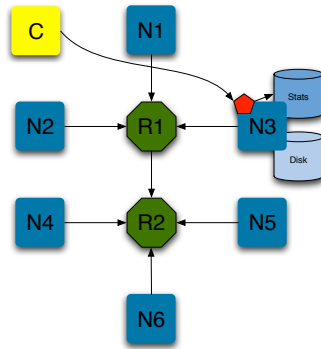


Figure 3-1: BBUStatistics - gathers informations related to the queries that arrived on a node.

decide to replicate also all or some of its children.

- *Replication based on node load* - if a node detects that it is too busy serving popular content it will request help from a new node, this way the load will be balanced.

In this chapter we have described replication strategies used by different peer-to-peer applications. We have also introduced a new *location aware* replication strategy for the BBUFs peer-to-peer system. This strategy uses different input sources, like:

- Round trip times.
- Time to live.
- GPS coordinates in order to choose a good replica candidate.

Chapter 4

Formal Model

4.1 Π -calculus

Over the last decades formal models have been successfully employed to specify communication protocols or systems and to verify their properties. A verification process can be viewed in two steps: specifying a protocol/system (and so, providing a model), and verifying these properties. A formal specification is the definition of an entity, an object or class, using a description technique [Cho85]. Using mobile ambients [CGG99] or other formalisms that are extending them, like Timed Mobile Ambients for Network Protocols, we can easily express common network protocols. In this chapter we concentrate on the process algebraic technique for expressing the lookup mechanism.

Other formal approaches, like assertional proof method, has been used for the problem of concurrent topologies in peer-to-peer networks [LMP04]. Krishnamurthy et al. [KEAAH05] present an analytical study of Chord under churn using a master-equation-based approach. Bakhshi and Gurov used a π -calculus approach to formalize the Chord system [SMK⁺01]. They even verified the correctness of Chord's stabilization algorithm by establishing weak bisimulation between the specification of Chord as a ring network and the implementation of the stabilization algorithm, both modeled in the π -calculus [BG07].

Using process algebra we can express the behavior of a system in a systematic,

precise and modular way. The basic building blocks are processes and channels. Process algebra is compositional, providing scalability of the formal specification by enabling the composition of large processes using smaller ones. We refer to process algebras like CCS [Mil99], and π -calculus [Par01].

Widely used to model dynamic and interacting systems, the π -calculus allows to express the mobility and the connectivity among the represented processes. By allowing to pass channels as data along other channels it is able to easily express complex processes. The mobility is a feature that helps to express easily certain properties that are difficult to describe in formalisms without mobility. Having a simple semantics it is an attractable formalism for expressing network processes.

We present in this section the monadic version of the π -calculus, meaning that a message consists of exactly one name. We consider χ to be an infinite set of names. The elements of χ are denoted by x, y, z, \dots . The terms of this formalism are called processes and each process is denoted by P, Q, R, \dots .

Definition 1. The **processes** are defined over the set χ of names by the following grammar:

$$P ::= 0 \mid \bar{x}\langle z \rangle.P \mid x(y).P \mid P|Q \mid P + Q \mid !P \mid \nu xP$$

The process expressions are defined by guarded processes $\bar{x}\langle z \rangle.P$ and $x(y).P$, parallel composition $P|Q$, nondeterministic choice $P + Q$, replication $!P$ and a restriction νxP creating a new channel x for the process P . 0 has a special meaning, representing the empty process. The guards represent sending (output guard) and receiving (input guard) a message along a channel.

A structural congruence relation is defined over the set of processes. We denote by $fn(P)$ the set of the names with free occurrences in P , and by $=_\alpha$ the standard α -conversion.

Definition 2. The relation \equiv over the set of processes is called **structural congruence**, and it is defined as the smallest congruence which satisfies:

- $P \equiv Q$ if $P =_\alpha Q$,

- $P + 0 \equiv P$, $P + Q \equiv Q + P$, $(P + Q) + R \equiv P + (Q + R)$,
- $P|0 \equiv P$, $P|Q \equiv Q|P$, $(P|Q)|R \equiv P|(Q|R)$,
- $!P \equiv P|!P$
- $\nu x 0 \equiv 0$, $\nu x \nu y P \equiv \nu y \nu x P$, $\nu x (P|Q) \equiv P|\nu x Q$ if $x \notin fn(P)$.

Structural congruence deals with aspects related to the structure of the processes. The evolution of a process is described in the π -calculus by a reduction relation over processes called reaction. This reaction relation contains those transitions which can be inferred from a set of rules.

Definition 3. The **reduction relation** over processes is defined as the smallest relation \rightarrow satisfying the following rules:

$$\text{(com)} \quad (\bar{x}(z).P + R_1)|(x(y).Q + R_2) \rightarrow P|Q\{z/y\}$$

$$\text{(par)} \quad P \rightarrow Q \text{ implies } P|R \rightarrow Q|R$$

$$\text{(res)} \quad P \rightarrow Q \text{ implies } (\nu x)P \rightarrow (\nu x)Q$$

$$\text{(str)} \quad P \equiv P', P' \rightarrow Q' \text{ and } Q' \equiv Q \text{ implies } P \rightarrow Q$$

The most studied forms of behavior equivalence in process algebras are based on the notion of bisimulation. Among the definitions of bisimilarity the open bisimulation is given by using the labeled transition system defined by the reduction rules. By studying the bisimilarity between two processes the systems can be checked automatically. *Weak open bisimilarity* allows a basic verification technique for proving properties about mobile concurrent systems [VM94].

In Section 2.4.3 we present the lookup mechanism and describes the protocol using π -calculus [CC14]. Also the Mobile Workbench agents and the verification results are presented [VM94].

4.1.1 Lookup Details

The π -calculus Expressions for the Lookup Mechanism

We define $Lookup(memory, network, request, response, failed, valid, address, read, console, exists, missing, rule, verifier, check, message)$ as:

$$\begin{aligned} Lookup(memory, \dots) &\stackrel{def}{=} memory(inputParam). \\ &\quad \overline{network}\langle request \rangle. Medium(\dots) \\ &\quad network(response). ProcessResponse(\dots) \end{aligned}$$

The *Medium* process is more complex since it is able to communicate both with the client and the system nodes, nodes that are able to serve the client requests.

$$\begin{aligned} Medium(\dots) &\stackrel{def}{=} memory1(message). \overline{verifier}\langle rule1 \rangle. \\ &\quad check1(rule1). ([rule1 = exists1] Router(network1, \dots) + \\ &\quad [rule1 = missing1](memory2(message). \overline{verifier}\langle rule2 \rangle. \\ &\quad check2(rule2). ([rule2 = exists2] Router(network2, \dots) + \\ &\quad [rule2 = missing2] Medium(\dots)))) \end{aligned}$$

The *Node* is the process that is receiving the request via the network channel from the *Medium*.

We define $Node(network, analyze, memory, screen, unicast, message, response, result, valid, failed)$ as:

$$\begin{aligned} Node(network, \dots) &\stackrel{def}{=} network(message). \overline{analyze}\langle message \rangle. \\ &\quad analyze(response). ([response = failed] FailedReply(network, \dots) \\ &\quad + [response = valid] SuccessReply(network, \dots)) \end{aligned}$$

This way we are able to express the interaction between the components that are involved in the model. But we can notice that *time*, a key attribute in network communication, is missing. Using only π -calculus we are not able to express what is

happening if a channel is interrupted, or when a subprocess is not able to respond in a timely fashion. By using the timed extension proposed by Berger et al. in [BH03] we can express how the system reacts in such cases too.

4.1.2 Timed π -calculus

Redefining With Timers

Since the communication is maintained using IP protocols we are using the following timeouts:

- *SO_RCVTIMEO* by *ProcessResponse* sub-process, since the lookup is using User Datagram Protocol (UDP) to perform the request and wait for a response.
- *TIME_WAIT* by *Connect* sub-process, since once the lookup response is received it is connecting using Transmission Control Protocol (TCP) to establish and maintain the connection.

We are using the User Datagram Protocol (UDP) [Pos80], part of the IP protocol suite, for the lookup operation. Because of the decentralized nature of the system, the client is aware that the nodes, that are holding the requested content, may be unavailable [Coj11]. Also because we are maintaining multiple number of replicas it is not important what node is serving the client request [CB09a].

All the subprocesses mentioned in Section 4.1.1 that are using the network channel should be redefined with timers. The same definitions, presented in the Section 4.1.1, are used. We are redefining only the main sub-processes that are performing time aware operations, *ProcessResponse* and *Connect*:

$$\begin{aligned}
& \text{ProcessResponse}(\text{network}, \dots) \stackrel{\text{def}}{=} \text{memory}(\text{response}). \\
& \quad ([\text{response} = \text{failed}] \text{timer}^{SO_RCVTIMEO} \\
& \quad (0, \text{memory}(\text{failedresponse}).\text{FailedResponse}(\dots)) \\
& \quad + [\text{result} = \text{valid}] \text{memory}(\text{successresponse}). \\
& \quad \text{SuccessfulResponse}(\text{memory}, \dots)) \\
& \text{Connect}(\text{memory}, \dots) \stackrel{\text{def}}{=} \text{memory}(\text{request}).\overline{\text{network}}\langle \text{request} \rangle. \\
& \quad \text{network}(\text{result}).\overline{\text{memory}}\langle \text{result} \rangle.\text{memory}(\text{result}). \\
& \quad ([\text{result} = \text{failed}] \text{timer}^{TIME_WAIT} \\
& \quad (0, \text{FailedResponse}(\text{memory}, \dots)) + \\
& \quad [\text{result} = \text{valid}] \text{Connect}(\text{memory}, \dots))
\end{aligned}$$

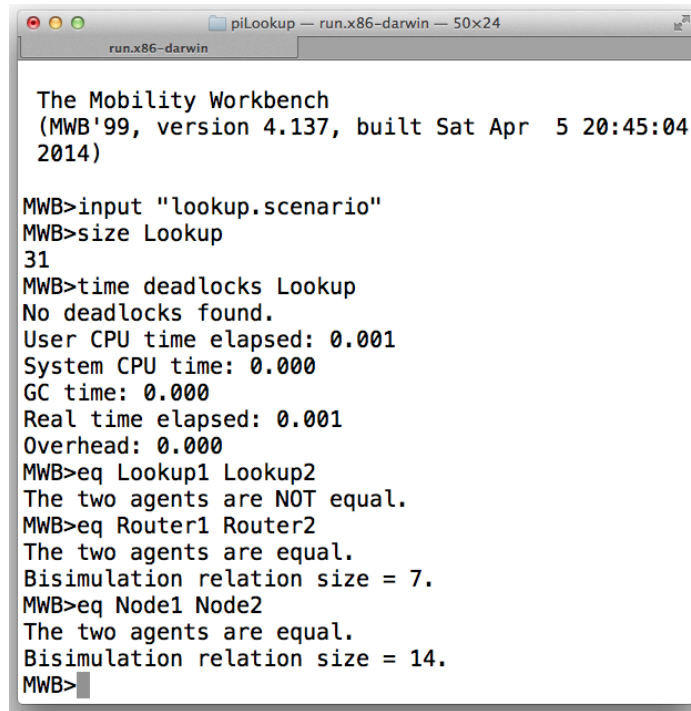
4.1.3 Model Validation

Using MWB we are able to define and validate the Lookup model presented in Section 4.1.1.

Using the observational equivalence (bisimulation) we can compare and see if two agents are the same. From Figure 4-1 we can see that *Lookup1* and *Lookup2* are different, since are triggered from different network locations. Also, when comparing *Node1* with *Node2* and *Router1* and *Router2* we can see that they are having the same behavior.

4.2 Mobile Ambients

Ambient calculus was introduced by Cardelli and Gordon in 1999 as a need to express the movement of processes or devices, even between different administrative domains [CGG99]. The ambient calculus differs from other formalisms such as π -calculus [Mil99] in such that the computational model is based on *movement* instead of *communication*, like we saw in Section 4.1.1.



```
run.x86-darwin piLookup — run.x86-darwin — 50x24
run.x86-darwin
The Mobility Workbench
(MWB'99, version 4.137, built Sat Apr 5 20:45:04
2014)

MWB>input "lookup.scenario"
MWB>size Lookup
31
MWB>time deadlocks Lookup
No deadlocks found.
User CPU time elapsed: 0.001
System CPU time: 0.000
GC time: 0.000
Real time elapsed: 0.001
Overhead: 0.000
MWB>eq Lookup1 Lookup2
The two agents are NOT equal.
MWB>eq Router1 Router2
The two agents are equal.
Bisimulation relation size = 7.
MWB>eq Node1 Node2
The two agents are equal.
Bisimulation relation size = 14.
MWB>
```

Figure 4-1: Verificarea mecanismului

In this section we extend the use of composition and choice operations, that were defined in [CGG99] for process operations, to be used on *domain* attribute also. Using these two operations we are able to express routing and addressing scheme like: unicast, anycast and multicast.

4.2.1 Base Concepts

cMa Formal Syntax

The following table describes the syntax of coordinated mobile ambients.

4.2.2 Domain Mobile Ambients Behavior

In this section we present our proposal to extend the model presented in [Cio09]. By overloading the domain attribute, in order to enhance the local knowledge of an ambient, we can encapsulate the domain choice of an action. We will be able to express or formalize all sorts of actions and mechanism, like: the choices that a student has

Table 4.1: Coordinated Mobile Ambient Syntax

n,m,p	ambient names	P,Q ::=	processes
C	::=	capabilities	0
in n	can enter n	C.P	movement
out n	can exit n	$(n_{(l,h,d)}^{\Delta t}[P], Q)$	ambient
open n	can open n	P Q	composition
go y	migration to y	P+Q	choice
		$M^{\Delta t}.(P, Q)$	movement
		(vn)P	restriction
		*P	replication

when leaving the university location. Also we could easily formalize TCP/IP unicast, anycast, multicast and broadcast routing schemes. Since the differences between the routing schemes are primary on the endpoint levels, we propose to add composition and choice operations to the domain attribute to be able to formalize all the defined routing schemes.

4.2.3 Extending Domain Attribute of an Mobile Ambient

To represent an endpoint group, we use the notation d_m^n , where:

- m - represents the total number of nodes in the domain.
- n - represents the number of nodes that will be chosen by the routing scheme.

Using the above notations for each routing scheme we can define the following endpoint groups.

- *unicast group* = $d_1^1 \equiv d_1$ - the client request will be routed to the single node that represents the unicast group.
- *anycast group* = $d_m^1 \equiv d^1$ - the client request will be routed to only one node that is part of the anycast group. Here the anycast group has m nodes, but in this case we will contact only a single node.

- *multicast group* = $d_m^n \equiv d^n$ - the request will be routed to n destinations from the total of m nodes of a network. n represents the total number of nodes in this multicast group. n being less than m .
- *broadcast group* = $d_m^m \equiv d_m$ - the request is routed to all the available nodes, n from the multicast group is equal to m .

Considering the above domain definitions we can define an ambient with an unicast domain in the following way:

$$(n_{(l,h,d_1)}^{\Delta t} [P], Q) \equiv (n_{(l,h,d_i)}^{\Delta t} [P], Q)$$

where i is one of the subdomains.

Also an ambient with anycast domain has the following form:

$$(n_{(l,h,d^1)}^{\Delta t} [P], Q) \equiv (n_{(l,h,d_1+d_2+\dots+d_m)}^{\Delta t} [P], Q)$$

and it will use choice operation on domain field.

Likewise an ambient with multicast or broadcast domain will be represented as follows:

$$(n_{(l,h,d^n)}^{\Delta t} [P], Q) \equiv (n_{(l,h,d_1|d_2|\dots|d_n)}^{\Delta t} [P], Q)$$

and it will use composition on domain field.

In this chapter we presented two ways to express the Lookup mechanism. Also using the proposed changes we are able to write more concise and compact expressions when describing complex network algorithms and the need of multiple domain choices are involved. Having a concise way to express the movement is helping in validating and comparing classic routing algorithms on all type of networks, independent of the used addressing scheme.

The formalization and the verification of the network protocols was and still is a hot research subject since it is allowing us to validate new or existing models.

Conclusions

The purpose of this thesis is to highlight that the decentralized unstructured distributed file systems are an important part of the system operation research field. File systems are used for more than 40 years already and in all this time they have grown into bigger and complex systems. The latest trend being to migrate all the data in cloud. The cloud being the name of multiple complex systems that are containing the file system also. Usually scattered over multiple continents in order to offer a high performance to their users.

In this paper we presented the main results from the decentralized distributed file system domain, the context that made them possible and the novelty that they are offering. The thesis is presenting new approaches, that such systems are able to use, a new way to lookup data based on IPv6, a new way to load balance user queries and how to replicate the content. Also a new way to select replication destinations. The formalization of such mechanism was another topic that we have addressed. By extending the Timed Coordinated Mobile Ambients model we were able to express mechanism like: anycast, multicast and broadcast.

For each approach that we introduced we suggested also some implementations, that could expand the research in the decentralized unstructured file system field.

For further forks we are proposing to extend the models suggested, to improve the proposed approaches and to present new ways to enhance the field to cope with the client demands.

Bibliography

- [Ber03] J. E. Berkes. Decentralized peer-to-peer network architecture: Gnutella and freenet, 2003.
- [BG07] Rana Bakhshi and Dilian Gurov. Verification of peer-to-peer algorithms: A case study. *Electronic Notes in Theoretical Computer Science*, 181:35–47, 2007.
- [BH03] Martin Berger and Kohei Honda. The two-phase commitment protocol in an extended π -calculus. *Electronic Notes in Theoretical Computer Science*, 39(1):21–46, 2003.
- [BKK⁺03] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in p2p systems. *Commun. ACM*, 46(2):43–48, 2003.
- [BPS05] Jean-Michel Busca, Fabio Picconi, and Pierre Sens. Pastis: A highly-scalable multi-user peer-to-peer file system. In Jos C. Cunha and Pedro D. Medeiros, editors, *Euro-Par*, volume 3648 of *Lecture Notes in Computer Science*, pages 1173–1182. Springer, 2005.
- [CB09a] Dan Cojocar and Florian Mircea Boian. BBUFs: Replication strategies. In *Proceedings of Knowledge Engineering: Principles and Techniques Conference (KEPT)*, pages 284–287, Cluj–Napoca, Romania, July 2009.
- [CB09b] Dan Cojocar and Florian Mircea Boian. BBUFs: Replication Strategies Comparison. *KEPT2009 Knowledge Engineering Principles and Techniques*, Selected Papers:343–350, 2009.
- [CC14] Gabriel Ciobanu and Dan Cojocar. Expressing BBUFs lookup using the π -calculus. In *Workshosp on Global Computing Models and Technologies, co-located with SYNASC 2014*, 2014.
- [CCC⁺05] Chi-Yuan Chang, Wei-Ming Chen, Han-Chieh Chao, T. G. Tsuei, and Hong Bin Liu. Ip layer load balance using fuzzy logic under ipv6 anycast mechanism. *Int. J. Netw. Manag.*, 15(5):311–319, 2005.
- [CDG⁺08] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and

- Robert E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2):4:1–4:26, June 2008.
- [CGG99] Luca Cardelli, Andrew D. Gordon, and Giorgio Ghelli. Mobility types for mobile ambients. In *ICAL '99: Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, pages 230–239, London, UK, 1999. Springer-Verlag.
- [Cho85] Tat Choi. Formal techniques for the specification, verification and construction of communication protocols. *Communications Magazine, IEEE*, 23(10):46–52, October 1985.
- [Cio09] Gabriel Ciobanu. Coordinated mobile agents. In *Proceedings of the Romanian Academy, Series A, Volume 10, Number 1*, 2009.
- [Coj08a] Dan Cojocar. BBUFs: A new lookup mechanism based on IPV6. In *Workshosp on Global Computing Models and Technologies, co-located with SYNASC 2008*, pages 358–361, 2008.
- [Coj08b] Dan Cojocar. BBUFs: Synchronization mechanism. In *6th International Conference of Applied Mathematics (ICAM)*, pages 363–368, 2008.
- [Coj09] Dan Cojocar. The Architecture of BBUFs. *KEPT2009 Knowledge Engineering Principles and Techniques*, Selected Papers:335–342, 2009.
- [Coj10] Dan Cojocar. Bbufs: Routing protocol. In *Proceedings of the Symposium Colocviul Academic Clujean de Informatica*, pages 116–121, Cluj–Napoca, Romania, June 2010.
- [Coj11] Dan Cojocar. Replication location decisions. In *Nano, Information Technology and Reliability (NASNIT), 2011 15th North-East Asia Symposium*, pages 161–165, Macao, China, October 2011.
- [Coj12] Dan Cojocar. Load balance queries in decentralized peer-to-peer file systems. In *Proceedings of the National Symposium ZAC2012 (Zilele Academice Clujene, 2012)*, pages 105–110, 2012.
- [CSWH01] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46–63, 2001.
- [DH98] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946.
- [DKK⁺01] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with cfs. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 202–215, New York, NY, USA, 2001. ACM.

- [EJ01] Donald Eastlake and Paul Jones. Us secure hash algorithm 1 (sha1), 2001.
- [HGM10] Keith B Hall, Scott Gilpin, and Gideon Mann. Mapreduce / bigtable for distributed optimization. *Neural Information Processing Systems Workshop on Learning on Cores Clusters and Clouds*, 1(1):1–7, 2010.
- [HKM⁺88] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and performance in a distributed file system. *ACM Trans. Comput. Syst.*, 6(1):51–81, 1988.
- [HSW94] Yixiu Huang, Prasad Sistla, and Ouri Wolfson. Data replication for mobile computers. *SIGMOD Rec.*, 23(2):13–24, May 1994.
- [Jos03] S Josefsson. Rfc 3548-the base16, base32, and base64 data encodings. *Internet Request for Comments*, 2003.
- [KBC⁺00] John Kubiawicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. Oceanstore: an architecture for global-scale persistent storage. *SIGPLAN Not.*, 35(11):190–201, 2000.
- [KEAAH05] Supriya Krishnamurthy, Sameh El-Ansary, Erik Aurell, and Seif Haridi. A statistical theory of chord under churn. In *Peer-to-Peer Systems IV*, pages 93–103. Springer, 2005.
- [LCC⁺02a] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, ICS '02, pages 84–95, New York, NY, USA, 2002. ACM.
- [LCC⁺02b] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, pages 84–95. ACM, 2002.
- [LM09] Avinash Lakshman and Prashant Malik. Cassandra: structured storage system on a p2p network. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 5–5. ACM, 2009.
- [LMP04] Xiaozhou Li, Jayadev Misra, and C Greg Plaxton. Brief announcement: Concurrent maintenance of rings. In *PODC*, page 376, 2004.
- [LNS96] W Litwin, M A Neimat, and D A Schneider. LH* - A scalable, distributed data structure. *ACM Transactions on Database Systems*, 21(4):480–525, 1996.

- [LSSD02] Houda Lamahemedi, Boleslaw Szymanski, Zujun Shentu, and Ewa Deelman. Data replication strategies in grid environments. In *in Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02)*, pages 378–383. Press, 2002.
- [Mil99] Robin Milner. *Communicating and mobile systems: the π -calculus*. Cambridge University Press, New York, NY, USA, 1999.
- [MK03] Robert W Moss and Peter Korger. Methods and structure for read data synchronization with minimal latency, November 11 2003. US Patent 6,646,929.
- [MMGC02] Athicha Muthitacharoen, Robert Morris, Thomer M. Gil, and Benjie Chen. Ivy: a read/write peer-to-peer file system. *SIGOPS Oper. Syst. Rev.*, 36(SI):31–44, 2002.
- [MMP94] Drew Major, Greg Minshall, and Kyle Powell. An overview of the network operating system. In *WTEC'94: Proceedings of the USENIX Winter 1994 Technical Conference*, pages 27–27, Berkeley, CA, USA, 1994. USENIX Association.
- [NWO88] Michael N. Nelson, Brent B. Welch, and John K. Ousterhout. Caching in the Sprite network file system. *ACM Transactions on Computer Systems*, 6(1):134–154, 1988.
- [Ora01] Andrew Oram. *Peer-to-peer: harnessing the benefits of a disruptive technology*. " O'Reilly Media, Inc.", 2001.
- [Par01] Joachim Parrow. An introduction to the-calculus. *Handbook of Process Algebra*, pages 479–543, 2001.
- [PC04] Franjo Plavec and Tomasz Czajkowski. Distributed File Replication System based on FreePastry DHT. Technical report, University of Toronto, Ontario, Canada, 2004.
- [Pos80] J Postel. Rfc 768: User datagram protocol (udp). *Request for Comments, IETF*, 1980.
- [Pos81] J. Postel. Internet Protocol. RFC 791 (INTERNET STANDARD), September 1981. Updated by RFCs 1349, 2474, 6864.
- [PRR97] C. Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, 1997.
- [RD01] Antony Rowstron and Peter Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. *SIGOPS Oper. Syst. Rev.*, 35:188–201, October 2001.

- [SGK⁺85] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. Design and implementation of the Sun Network Filesystem. In *Proc. Summer 1985 USENIX Conf.*, pages 119–130, Portland OR (USA), 1985.
- [SKRC10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.
- [SMK⁺01] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.
- [Tra95] P Traina. Bgp-4 protocol analysis. 1995.
- [VM94] Björn Victor and Faron Moller. The Mobility Workbench — a tool for the π -calculus. In David Dill, editor, *CAV'94: Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Springer-Verlag, 1994.
- [WJH97] Ouri Wolfson, Sushil Jajodia, and Yixiu Huang. An adaptive data replication algorithm. *ACM Trans. Database Syst.*, 22(2):255–314, June 1997.
- [XJZZ00] Dong Xuan, Weijia Jia, Wei Zhao, and Hongwen Zhu. A routing protocol for anycast messages. *Parallel and Distributed Systems, IEEE Transactions on*, 11(6):571–588, 2000.
- [ZnWLY11] Han Zhi-nan, Yan Wei, Zhang Li, and Wang Yue. Design and implementation of an anycast efficient qos routing on ospfv3. 2011.