Management and semantic description of objects for the future internet

Eric Renault, Wassim Drira, Houssem Medhioub, Djamal Zeghlache

Management and Semantic Description of Objects for the Future Internet

Éric Renault, Wassim Drira, Houssem Medhioub and Djamal Zeghlache
Institut Télécom – Télécom SudParis
Samovar UMR INT-CNRS 5157
Évry, France

***Abstract*— By moving from its original host-centric architecture to a new information-centric organization, the Internet will be able to offer new services and applications to end users, allowing for example the on-demand composition of a new service from those already available online. However, this requires the development of a Network of Information to offer users the possibility to annotate, discover and access digital and real-world objects in a convenient way. As such, this paper presents an original architecture to support these new services based on three underlying spaces: a storage space to keep data resilient, an index space to allow efficient search and a communication space to ensure an efficient transfer of data between the different entities.**

INTRODUCTION

With about 1.5 billion people connected on a daily basis and more than 4 billion expected over the next few years, the Internet has become one of the most strategic infrastructure for more than a decade with a key socio-economical role leading innovation, economic growth and productivity at a world-wide scale. This has been made possible with the development of new technologies including wireless infrastructures involving anywhere and anytime connectivity together with the emergence of communicating objects like RFID tags or real and virtual world objects, this moving the current Internet toward an Internet of Things. New usages will appear and applications will be significantly different. Many social-economical branches will take benefits of this new services like health care, education, energy management or proximity services. However, this will become feasible only if efficient solutions are provided to describe and access digital and real-world objects. Organizations and institutions all around the world are funding research and development projects to design a new Internet that shall meet these new needs and demands.

One way to achieve this goal is to move the current *Host-centric* organization of the Internet (based on a set of servers that users are expected to get connect to) to an *Information-centric* organization where users can focus on the description of their digital and real-world objects rather than the way to access them. This involves the development of a *Network of Information*, i.e. a high-level virtual architecture that offers the possibility to describe, index, search, manage and transparently access objects.

This article presents the design of an original architecture to support the description and the management of objects for the Future Internet. After introducing both the object model and the information model developed in the scope of the Network of Information, the article first focusses on the global architecture to show how the Future Internet could be divided up into different spaces for storing, indexing and communicating. Then, a detailed description of both index and storage spaces is provided followed by the way the Network of Information could be implemented on top of it. The last part of the paper is devoted to the presentation of the research directions we will pursue in the near future.

I. OBJECT AND INFORMATION MODELS

The work presented here has been built based on two complementary models: on the one hand, the object model aims at specifying the different kinds of objects managed by the Network of Information; on the other hand, the information model aims at allowing the description of objects provided by end-users and applications. These two models are just provided for reference to the reader as more information are available at [2] and [3] for the object model and the information model respectively.

A. Object model

The Network of Information is able to manage two different types of objects: Information Objects on the one hand and Bit-level Objects on the other hand. Fig. 1 shows the relationship between the objects.

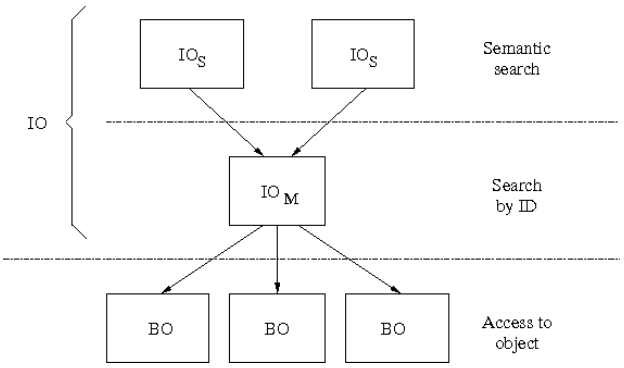

Fig. 1. The object model.

A *Bit-level Object* (BO) is the digital representation of the associated object. From the Network-of-Information point-of-view, this is just a set of bits which meaning is only relevant

to the end-user and/or application. From the user perspective, a Bit-level Object is typically a content like a web page, a sound track, a movie, etc. It can also be a mean to access non-digital objects like mobile phones or RFID tags or any other relevant device which may be interested to be connected to via the Fture Internet. In this case, the Bit-level Object can be a process the user can use to access the associated non-digital object.

An *Information Object* (IO) contains the information related to the object that are not part of the content. These information may be of two different kinds :

- a *Semantic Information Objects* (IO_S) stores the semantic description of an object, i.e. all the information the owner of an object considers useful to understand and/or search for the object.
- a *Management Information Object* (IO_M) stores the information associated to the object that are relevant for its management from the NetInf point of view.

If Semantic Information Objects are managed by users, Management Information Objects are managed by NetInf according to the information provided by their owners.

Some digital objects may grow to a very large size (this is typically the case of videos). As a result, more than one BO can be associated to an object. This approach is very similar to the concept of chunks as used in BitTorrent [4]. In the same way, more than one Information Object can be associated to a given set of BOs. There can be at the same time an IO for the semantic description of the object and another one for its management. Moreover, there can be concurrent semantic descriptions for a single object, for example if two descriptions are provided for two different kinds of users with two distinct access rights and/or point of view.

A connection to the current Internet can be made to understand BOs and IOs, and highlight the interest to separate them. A web page available on the Internet is mainly composed of the three following elements: the URI that indicates the location of the page, the content that is to be intrepreted by the web browser and a set of information used to describe the content of the page (the `meta` tags of the HTML language), this last element not being mandatory. In the scope of the Object model, the content of the web page is BO, the URI is the part of the IO_M and the `meta` tags are stored inside an IO_S.

B. Information model

Together with the object model as presented above, an Information model has been developed. The Information model is in charge of defining how Information Objects shall be presented to and stored in NetInf. The one developed in the scope of the Network of Information is called the Metalist Model, as it aims at including lists of metadata. The Metalist model includes a set of basic functionalities that allows an easy-to-use management of IOs, whenever it is a Semantic or a Management IO. Three ways have been identified to specify metadata in an Information Object using the Metadata Model:

- a metadata can be provided in the Information Object directly using the `metadata` label. In this case, the value can be any information. It is possible to tag the value using an `attribute` name. This is especially interesting both to allow semantic searches by the user and to help NetInf managing objects.
- a metadata (or a set of metadata) can be included from another pre-existing metalist. This functionality has been included in order to avoid users' redundancy, ie. from providing several times the same list of metadata for a set of objects (for example, it makes it possible the factorisation of the description of a set of pictures related to a single subject) and thus to increase the consistency of the description information.
- a metadata (or a set of metadata) can be included from external metadata, ie. metadata that are not in the metalist format. This can be the EXIF format [5] (the one used to store pictures' metadata in digital cameras) or any other as long as a source-to-source translator is available to automatically translate the metadata in the original format to the metalist format.

The prototype is being developed in the scope of the 4WARD project [1] and will include the implementation of the Metalist Model. XML was chosen as the support language. The description of the Metalist Model provided above in this section only aims at providing an idea of what functionalities the Information Model can provide. For further information, [3] presents the complete description of the Metalist Model.

II. Global Architecture

The architecture of the Network of Information is based on three distinct spaces (see Fig. 2):

- the *Storage space* aims at storing all digital information related to the Future Internet. This can be objects directly, assuming these objects are digital ones, or there can be the digital mean to access real-world objects, for example a process to access an RFID tag.
- the *Index space* aims at storing the IO_S provided by end users so that the search for objects is performed more efficiently. In order to do so, an IO_S is preprocessed when presented to the index space and only the preprocessed version of the IO_S is stored in the index space.
- the *Communication space* stores nothing (at least from the NetInf point of view). It aims at providing a convenient mechanism to let the Network of Information, the Index space, the Storage space and the end user exchanging data. It can be based on any kind of protocol as long as some basic functionalities like a PUT and a GET are available.

Note that if BOs are stored in the storage space, IOs are stored in both index and storage spaces using two different formats. Before being stored in the index space, IOs are processed in order to make them easier to search for. As such, IOs are not stored literaly in the index space. Therefore, IOs are also stored in the storage space in order to allow end users

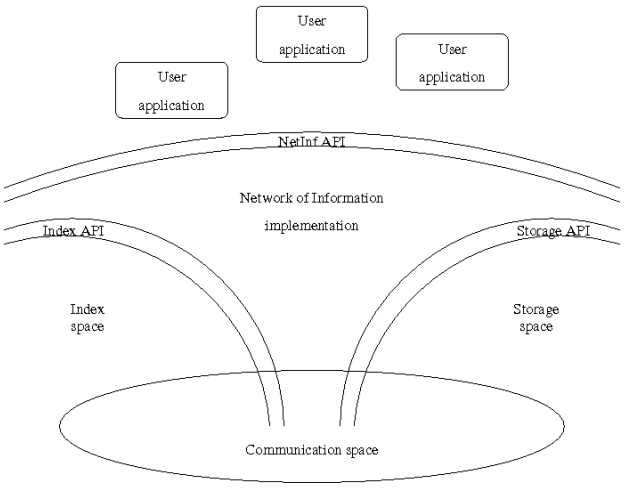

Fig. 2. NetInf global architecture.

to refer to their description (for example to perform updates if they do not have any personal storage device).

III. Design and Implementation

Considering the fast development of new technologies for storing, indexing and communicating, it is important to be able to adapt. As a result, the implementation of NetInf is not tightly coupled to these underlying spaces. Instead, a set of basic functionalities that must be available in any tools that would support either the index, the storage or the communication space of NetInf have been identified.

In the following, only operations for both storage and index spaces are specify. The communication space is assumed to be TCP/IP. However, alternative communication spaces like Generic Path [8] (the generic communication mechanism developed in the scope of the 4WARD project) will be taken into consideration in the future.

All operations presented in this section may return an error code independently from their ability to return a value or not. If no specific value should be returned by an operation, special value OK indicates that no error occurred while processing. In the following descriptions, no reference is made to errors as the possibility for operations to return an error is implicit.

A. Storage space

The NetInf storage space aims at offering persistency to Future Internet objects. These objects may be of two types: on the one hand, BOs are storing the effective digital representation of objects whether they are digital or real objects; on the other hand, IOs are storing meta-information about the objects themselves (these meta-information being managed by NetInf or the user himself). However, whatever the type of the object (IO or BO), they are identified by an ID.

Note that depending upon the nature of the underlying effective storage space, BOs presented to the storage space may either be complete objects or parts of objects after the original object has been divided up into *chunks*[1]. The ability for both NetInf and the storage space to divide digital objects into independant sub-parts may be of great help to optimize performance when accessing objects.

Put (IO | BO) → ID | Error: This function aims at storing an information object or a bit-level object. The value returned by the function is the ID that has been associated to the object.

Get (ID) → IO | BO | Error: This is the opposite function to the previous, i.e. it aims at retrieving the object that was previously stored in the storage space. The ID previously returned by the call to Put to store the object is provided as a parameter and the object is returned to the caller.

Update (ID, IO | BO) → OK | Error: The purpose of the Update function is to modify the content of the object. The two information required here are the identifier that was returned by Put at the creation of the object and the new content for the object.

Remove (ID) → OK | Error: The last operation that can be performed on the storage space is the removal of an object. In this case, the ID of the object is the only information required.

B. Index space

The goal of the NetInf index space is to preprocess object descriptions to efficiently answer semantic users' requests. Descriptions of objects are not necessarily stored as is in the index space. There may be stored; however, this is not a prerequisite. As a result, an IO_S like any other types of object must be stored in the storage space.

Put (IO_S, ID) → OK | Error: Operation Put aims at feeding the index space with a new description. The first parameter is the description of the object; the second parameter is the ID of the object that holds the description in the storage space. This is necessary to later return to users the description as it was initially provided (and not after it has been preprocessed by the index space).

Get (Request) → {ID} | Error: The second operation proposed for the index space aims at performing a request on the set of available descriptions. The caller must provide a request that specifies the characteristics of the object(s) it is looking for. The value returned by the function is the set of description IDs that matches the set of criteria. Note that several languages like XQuery [6] and TQL [7] are already available to specify requests on a set of XML nodes.

Update (ID, IO_S) → OK | Error: This function can be used to update an object description. Both the description object ID and the object description are required.

Remove (ID) → OK | Error: The purpose of the last operation available on the index space is removing an object description. This is performed after specifying the ID of the description object.

[1]The term chunk is used here as a reference to BitTorrent for the better understanding of the reader but no name has been officially specified in the scope of the 4WARD project.

C. Network of Information

The NetInf API is the only one that can be seen by end users and/or applications. As a result, it mainly aims at providing a uniform access to data and metadata stored in the underlying spaces.

Push (BO) → ID | Error (see Fig. 3): The goal of operation Push is to store a bit-level object. Upon the reception of the request, the object may be divided up into parts if it is too large. Then, the part(s) is (are) sent to the storage space. If there is any problem for any of the parts, the error code returned by the storage space is returned to the caller after the previously stored parts of the BO if any have been removed (for the purpose of clarity, this last step is not indicated on the diagram). After receiving the ID for all BOs, an IO_M is created and stored in the storage space. Upon success, the ID of the IO_M returned by the storage space is forwarded back to the caller and an error code is returned instead after removing everything in case of problem.

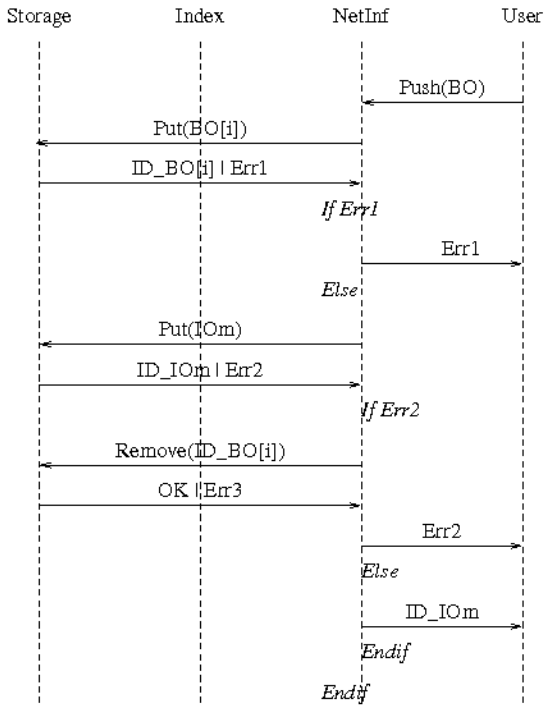

Fig. 3. Implementation of Push.

Get (ID) → BO | Error (see Fig. 4): Operation Get aims at retrieving the content of an object using its ID. The ID may belong to two different kinds of objects. On the one hand, the ID refers to an IO_S. In this case, the object is retrived from the storage space and returned to the caller. On the other hand, the ID necessarily refers to an IO_M. In order to be able to return the BO or the set of BOs to the caller, NetInf first need to get the content of the IO_M that stored the IDs associated to all the BOs. The bit-level objects requested to the storage space are then directly forwarded to the caller.

Publish (IO_S) → ID | Error (see Fig. 5): In the context of NetInf, publishing an object means that the description

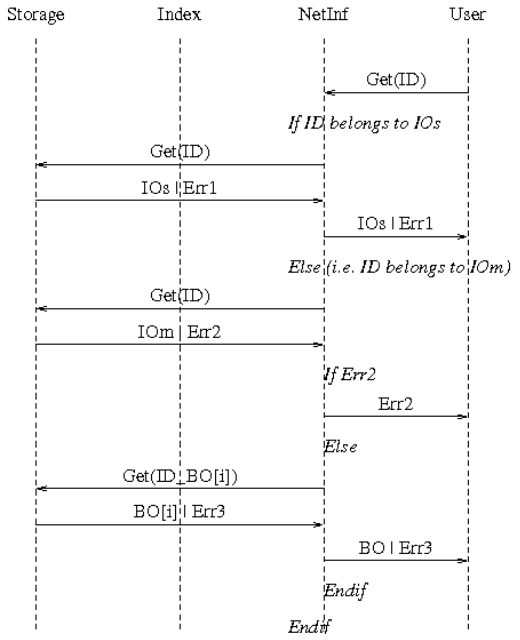

Fig. 4. Implementation of Get.

of the object is made available. As a result, the first step consists in storing the description in the storage space for further references and then ask the index space to take into account this new description.

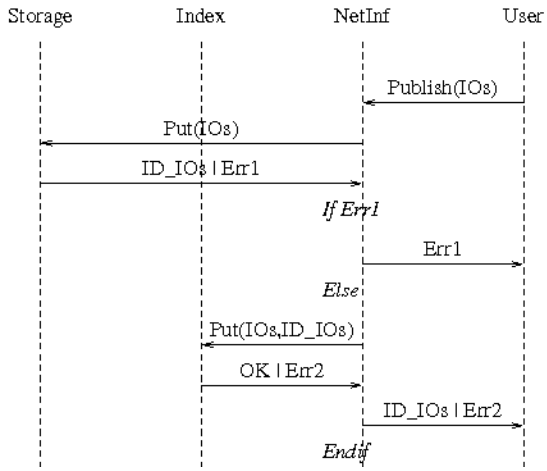

Fig. 5. Implementation of Publish.

Search (Request) → $\{IO_S\}$ | Error (see Fig. 6): The Search operation in the NetInf level is very similar to the Get operation at the index level. The only difference remains in the value returned by the two operations. While at the index space level the returned value is a list of IO_S IDs, the value returned by operation Search is the list of object descriptions. As a result, for each element of the list of IDs returned by the index space, a request is performed to the storage space in order to return to the caller the description object as provided by the user.

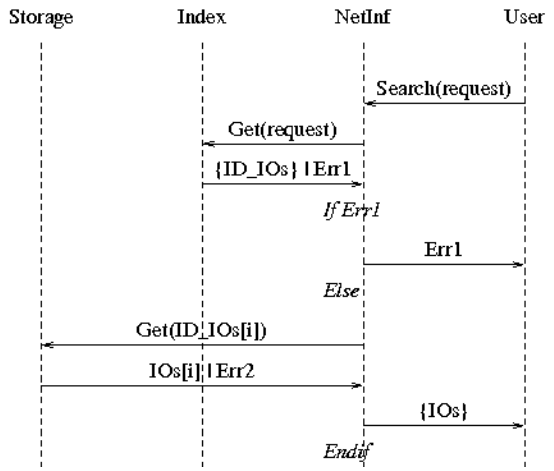

Fig. 6. Implementation of Search.

Update (ID, IO_S | BO) → OK | Error (see Fig. 7): The aim of the Update operation is typically to change the content of the object. However, this object can be of two different types. In any cases, the object stored in the storage space must be updated. Moreover, if the object is an IO_S then the published description must be updated too. This is of course performed if no error occurred when saving the object in the storage space.

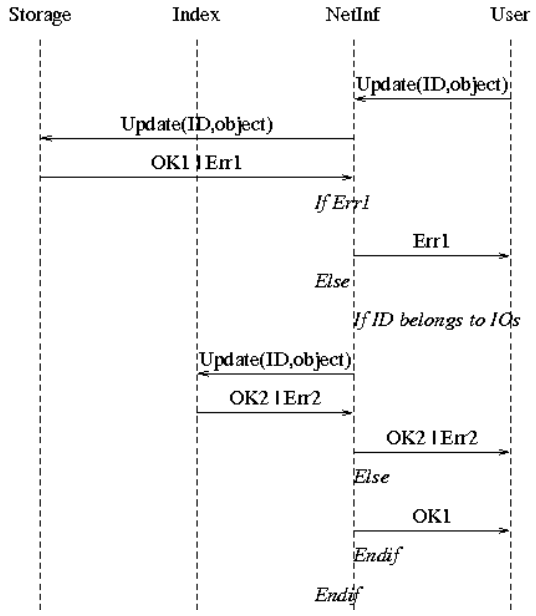

Fig. 7. Implementation of Update.

Remove (ID, IO_S | BO) → OK | Error (see Fig. 8): The last operation is once again the Remove operation. Like the Update operation, a distinction has to be made upon the type of the object. If the ID belongs to an IO_S, the object is first removed from the index space and upon success is then removed from the storage space. If the ID does not belong to an IO_S, this means that it belongs to an IO_M. In the latter case, the content of the object is first retrived in order to get the information to remove the associated BOs. After all BOs have been successfully removed, the object can finally be removed.

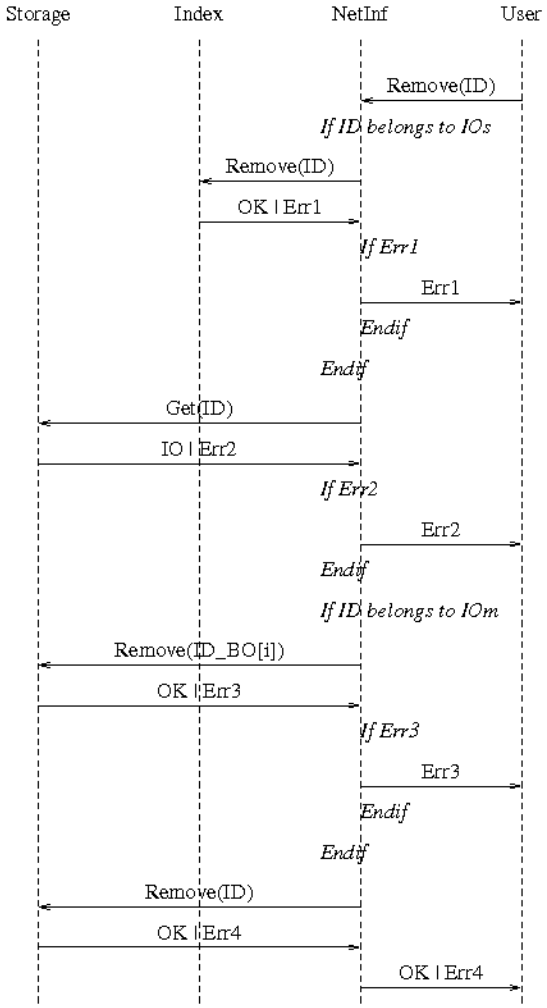

Fig. 8. Implementation of Remove.

IV. On going work

The NetInf architecture as defined above could be successfully implemented using various existing tools. For example, several solutions have been developed to store digital information at a global scale. Therefore, BitTorrent [4], OceanStore [9], Past [10], Pastry [11], PeerStore [12], Tahoe [13] or Venti-DHash [14] are examples that could be used to implement the storage space. In the same way, various tools have been developed to index XML files. Some interesting ones include BUS [15], eXist-db [16], Xindice [17] and XISS [18].

Apart from showing the feasability of this approach, the development of this framework is pursuing the following objectives:

- Evaluate the performance of the proposed architecture. The global response time for all the operations should

be very low as both indexing and searching XML native databases and the distributed storage of digital information have become very efficient over the past few years, and the overhead involved by the implementation of the both index space and storage space APIs is very low.

- Study the impact of security on the architecture. As presented above, any user can access any object (except if access rights have been provided at the storage space level). Therefore, some studies were initiated in order to identify the different kinds of access rights that should be implemented. Note that introducing access rights to this architecture is quite straightforward. In fact, any access to an object is performed through an IO_m. As a result, if granted access rights are stored in the IO_m, it becomes easier for NetInf to identify which users are allowed to access objects.
- Study the impact of mobility on the architecture. In the presentation above, it is considered that objects are static. What about moving objects like mobile phones or RFID tags. The location of these objects is a priority. This can be achieve by moving the ID associated to BOs to location information. However, the cost of such an operation, not in term of develpment, but in term of use shall be evaluated. Note that, similarly to the security case, the IO_m will be charge to store the mobility information.
- Study the impact of heterogeneity on the architecture. It has been assumed that a single index space and a single storage space were used. However, for many reasons that include security both to ensure redundancy and confidentiality, several index spaces and/or storage spaces could be used. Thanks to the use of the generic API defined to access both index and storage spaces, this should be quite straightforward to include. However, the cost or the gain in term of performance is to be identified.

V. Conclusion

The Internet as designed today is limited in terms of functionality. Especially, it limits the possibility to create large scale applications with automatic discovery due to its lacks of information about the available objects. The need for an alternative Internet that would include the description of objects together with the objects themselves has been clearly identified.

This paper proposed an architecture that merges both the ability to store objects at a very large scale, based on storage components that are already available and functional, and the ability to index these objects using efficient indexing tools. Moreover, the paper presented an abstraction of both storage and index spaces and showed how both spaces could cooperate throughout the Network of Information to allow semantic applications to end users.

For the near future, many studies are planed, based on the proposed architecture. After evaluating the performance, the impact of security, mobility and heterogeneity will be studied.

Disclaimer

This work has been supported by the IST 7th Framework Programme Integrated Project 4WARD, which is partially funded by the Commission of the European Union. The views expressed in this paper are solely those of the authors and do not necessarily represent the views of Institut Télécom (ex. GET-INT) or the respective projects and sponsors.

Short Bibliography

[1] *The FP7 4WARD Project*. http://www.4ward-project.eu/
[2] C. Dannewitz, K. Pentikousis, R. Rembarz, É. Renault, O. Strandberg and J. Ubillos. *Scenarios and Research Issues for a Network of Information*. MobiMedia'08, Oulu, Finland, July 2008.
[3] É. Renault and D. Zeghlache. *The Metalist Model: A Simple and Extensible Information Model for the Future Internet*. Proceedings of the 15th Open European Summer School and IFIP TC6.6 Workshop, EUNICE – The Internet of the Future, LNCS 5733, pp 88-98, Barcelona, Spain, September 2009.
[4] BitTorrent. http://www.bittorrent.com/
[5] *Exchangeable Image File Format for Digital Still Cameras: Exif Version 2.2*. Standard of Japan Electronics and Information Technology Industries Association, April 2002.
[6] XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery/
[7] L. Cardelli and G. Ghelli. *TQL: A query language for semistructured data based on the ambient logic*. Mathematical Structures in Computer Science, 14(3):285–327, Cambridge Journals, June 2004.
[8] H. Wösner and F. Guillemin. *The Generic Path: A Fresh View on Connectivity in the Future Internet*. 4WARD Workshop at ICT Mobile Summit, Stochholm, Sweden, June 2008.
[9] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, B. Zhao. *OceanStore: an architecture for global-scale persistent storage*. Proceedings of the ninth international conference on Architectural Support for Programming Languages and Operating Systems, pp 190–201, Cambridge, MA, November 2000.
[10] A. Rowstron and P. Druschel. *Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility*. ACM Symposium on Operating Systems Principles, pp 188–201, Banff, Canada, October 2001.
[11] A. Rowstron and P. Druschel. *Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems*. IFIP/ACM International Conference on Distributed Systems Platforms, pp 329–350, Heidelberg, Germany, November 2001.
[12] M. Landers, H. Zhang and K.-L. Tan. *PeerStore: Better Performance by Relaxing in Peer-to-Peer Backup*. Proceedings of the Fourth International Conference on Peer-to-Peer Computing, pp 72–79, Zurich, Switzerland, August 2004.
[13] Z. Wilcox-O'Hearn and B. Warner. *Tahoe: the least-authority filesystem*. Proceedings of the 4th ACM international workshop on Storage security and survivability, pp 21–26, Alexandria, VI, 2008.
[14] E. Sit, J. Cates and R. Cox. *A DHT-based Backup System*. Proceedings of the 1st IRIS Student Workshop, 2003.
[15] D. Shin, H. Jang and H. Jin. *BUS: An Effective Indexing and Retrieval Scheme in Structured Documents*. Proceedings of the 3rd ACM International Conference on Digital Libraries, pp 235–243, Pittsburgh, PA, June 1998.
[16] W. Meier. *eXist: An Open Source Native XML Database*. Web, web-services and database systems: NODe 2002 web- and database-related workshops, LNCS 2593, pp 169–183, Erfurt, Germany, October 2002.
[17] Apache Xindice. http://xml.apache.org/xindice/
[18] Q. Li and B. Moon. *Indexing and Querying XML Data for Regular Path Expressions*. Proceedings of the 27th International Conference on Very Large Data Bases, pp 361–370, Roma, Italy, September 2001.