On the need for a Control Mechanism in Generic Paths

S. Figueiredo*, Rui L. Aguiar*[¥]

***Abstract*— This paper tackles the need for a Resource Management Database under the scope of the Generic Path (GP) architecture, as a result of the 4WARD 7th FP project clean-slate approach. GP is a concept for generalizing data transport and/or transformation across a network facility, allowing for instance route identification and path classification. Though, the GP notion by itself misses a way to efficiently track availability of resources and for coordinating and combining the information of multiple GPs in a versatile way. The present work proposes filling such gap by adding these and other enhancements through the use of a hierarchical management solution, based on the intelligent organization and interaction of records. This approach is aimed to act as an adaptable and configurable resource management model, with concepts realizable in current networks by systematic deployment in network elements – therefore towards the notion of a flat architecture.**

***Index Terms*—Generic Path, resource management, cross-layer design, 4WARD**

I. INTRODUCTION

Continuous technology evolution has pushed Internet's range of services to be largely extended, both reflected in more obvious users' needs (e.g.: VoIP calls using IP signaling) or more disruptive paradigms (e.g.: social-networking as in Twitter). Bandwidth-hungry services such as Video on Demand (VoD) or Peer-to-Peer (P2P) raised network scalability and reliability problems not easily solved due to the extreme complexity derived from the existing "cocktail" of networking protocols. Another problem is that in most of the cases the introduction of a new application implies doing cross-design or layers patching, violating the Internet model. The global implementation of some possible solutions to the referred problems (and others like mobility), such as IPv6, multicast, IPSec, MIP or Quality of Service (QoS), have always faced reluctance by Service Providers in being widely provided, for example due to the lack of immediate gain or the possibility for a gradual alleviation of the problems through the natural evolution of technologies they were intended to solve [1].

* Instituto de Telecomunicações, Campus Universitário de Santiago, P-3810-193 AVEIRO - PORTUGAL
[¥] Universidade de Aveiro

In a novel vision of confronting the problem, the Clean-Slate approach proposes to totally redesign Internet architecture by ignoring current design constrains and analyzing actual and potential requirements. Theoretically, such an approach would permit mechanisms currently featured as add-ons to appear embedded in the base architecture. Initial work started at US through NSF's GENI (Global Environment for Networking Innovation) project for developing an infrastructure for supporting and implementing novel architectures [2], as part of its FIND (Future INternet Design) program. Two of those most known projects are RNA [3] and SILO [4]. RNA examines the implications of using a single, tunable protocol for different layers of the protocol stack, reusing basic protocol operations across different protocol layers. This avoids reimplementation, and encourages cleaner cross-layer interactions and support of dynamic service composition, and understanding the impact of layers in the architecture. As for SILO, it is characterized as a framework for highly configurable and complex communication tasks. It consists of building blocks that may be combined to accomplish from simple to highly complex communication tasks, and control elements to provide cross-layer interactions.

Following these steps, FIRE (Future Internet Research and Experimentation) program was launched under the 7th Framework Program of European Union, along with Japan's AKARI and many others from around the globe. The amount of work currently being developed bears proof of the importance of Internet's improvement for assuring world-wide sustainability [5].

The 7th FP 4WARD Project (which is officially complete) was the European's flagship for Future Internet research, combining multiple complementary network architectures in a common object-oriented framework [6]. Among its features (network of information, native management, virtualization and network of information which are out of scope in this paper), a new connectivity paradigm named GP is proposed. The GP architecture aims to reduce connectivity complexity by deploying a unique way to provide communication between any entity, regardless of their location or architectural level. In other words, it provides an abstraction for data transport across and/or data manipulation inside a network facility. Also, by its object-oriented approach, it allows the definition of classes for each path type and respective instantiation, as well as the identification of routes, among other features. Data transport over GPs occurs by mapping sessions into physical resources, such as links, network interfaces, routers, etc.

In order to facilitate network management and prevent today's problems such as lack of detailed network status information, silent failures or hidden dependencies, this paper describes a solution for unifying the way the path information is accessed and configured, the GP Resource Management Database.

To introduce our proposal, we first present the background of GP architecture in section II, dwelling in GP terminology and mechanisms. The paper then follows with a section on the GP Resource Management Database, spanning through the relationship between the different classes, the records organization within a node and a more practical example of the bootstrapping and basic API of the database. Additionally, a short reference is done about the benefits of having a well defined Resource Ontology supporting the GP resource management Database. A conclusion and possible future work closes the paper structure.

II. GENERIC PATH ARCHITECTURE

In order to generalize and abstract communication without being held by current Internet model, GP goals are [7]: (i) to develop foundations for both describing and prescribing any network communication; (ii) provide a generic communication service model not limited by any communication paradigm; (iii) design architectural constructs and primitives from this framework.

A s GP architecture follows an object-oriented design, its features include excursiveness, allowing description of a communication in a self-similar way; controlled opacity and virtualization, with state and functionality at any level accessible from any level; agnostic, relatively to technology, platform and communication; and modularity, providing the possibility to establish communication contexts and federations of network services, among others, such as polymorphism or overloading. The combination of these characteristics makes GP architecture a very powerful tool for network design and description, by selectively hiding the management complexity whenever desired.

Being an abstraction of current Internet architecture, GP is a way of looking at the network by taking advantage of properties deriving from its Object Oriented nature, allowing the creation of services not supported by the traditional network model. This explains the possibility for mapping GP terminology to existing entities and devices, observed throughout the rest of the paper. Thus, GP architecture is Clean-Slate in the sense that is allows the exploration of a new way to reimplement the protocol stack, again, following an object-oriented prism.

The entities that compose the presented architecture and their functions are described in the following subsection.

A. GP Architectural Elements

In this section, the terminology used within GP architecture is briefly described, though without neglecting relevant information for understanding the presented resource management solution:

Compartment – establishes the boundaries in which the communication exists.

Node Compartment (Node CT) – represents network or terminal nodes.

Entity - models a running service at any level and generalizes a communication data processing function. Entities exist within a Node CT and communicate with other Entities horizontally or vertically.

Generic Path (GP) – the central component and real innovation from GP architecture, it is the communication abstraction, and is represented by a horizontal connection between 2 Entities.

Hook – represents vertical communications between Entities within a same Node CT.

End Point (EP) – A GP is terminated by EPs, in the sense that an Entity is served by a GP through means of an EP. While Entities relate to the control and management of GPs (service discovery, routing, name resolution, etc), EPs pertain to data transfer and control (error control, flow control, encryption, coding, etc).

Mediation Point (MP) – aggregates/ interleaves multiple GPs, acting as the mediator between distinct GPs.

To clarify, and mapping to a typical process-based system like UNIX, an Entity abstracts a process, Hooks abstract inter process communication and Ports abstract process IDs / file descriptors. As for Compartments, they may map to a domain or a layer.

As referred, Entities establish GPs in order to communicate. The GP is the representation / abstraction of the communication, so it comprehends the set of necessary resources that provide end-to-end communication between two or more Entities. As the communication at any level is mapped to a (distinct) GP, Entities also exist at different communication levels (e.g. interpreting level as a layer, we may have UDP GPs or TCP GPs at transport CT, etc, with Entities acting as the Service Access Points in OSI model). So, with GPs mapping to any communication level, such as application or physical layer, the associated resources may be completely distinct. Referring to existing protocols, a Ethernet GP would have associated attributes such as Throughput, while a Radio GP class would have SNR and other related metrics and static properties. As a diversity of services may be represented by a GP, it needs to be contextualized: the scope of the communication is delimited by the Compartment (CT) in which the GP is running, and may for example correspond to a network, a protocol, an application, or procedure (local or remote); the GP type thus is associated with the CT in which it is running. The Node CT is a special (vertical) CT.

Another novel notion is that of End-to-End (E2E) GP, which represents the communication at its highest level of abstraction and allows a ubiquitous view of the path. This concept brings together the routing, resource , security and other attributes together in a single object, adding new possibilities to network design and management. It maps to the session the user is participating in, such as an audio conversation through VoIP, a P2P traffic transfer, or video-conference between multiple users. The E2E GP is composed from other GPs (through composition), each implementing a

different service technology. For instance, a VoIP call between 3 users would be represented by a single E2E GP, with that GP composed from (for example, considering a simple intra-domain connection) an UDP GP, 2 WiFi GP, a VoIP application GP, and all other GPs necessary for that session to be running.

All GPs requests are sent to a GP Factory (core elements, one per Node CT), which begins by checking for the existence of a previous GP to the same destination, and is responsible for node CT-internal information exchange, such as Entities instantiation for a particular CT. The CT works as an application framework, enhancing the GP with the GP class specific structure.

The interleaving of these elements is represented in Figure 1 (note: for the sake of simplicity, Hooks and Ports were not illustrated).

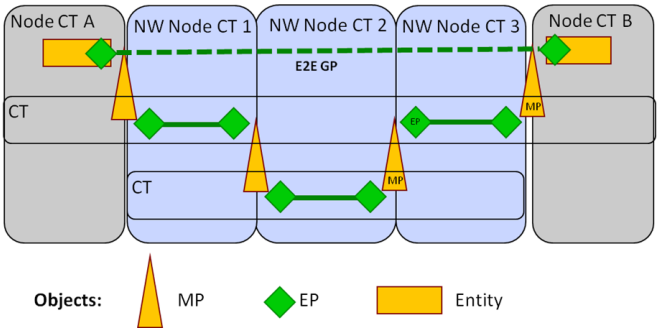

Figure 1 GP Architecture example

III. GP Resource Management Database

Under the GP Architecture, the traffic flow is ensured by the creation and management (modification, destruction) of multiple GPs. Each GP is characterized by the following GP Items: 1) associated EPs: 2 for unicast, more than 2 for multicast, anycast or broadcast; 2) sub-GPs: a GP being the composition of others; and 3) resources / attributes: QoS metrics, privacy settings, or any attribute relevant to the context under consideration. The access to the GP information is possible through Dials, for information inspection, typically state information, and Knobs, related to configurable information, such as QoS-related or security parameters.

The composable nature of GPs enables a superior resource management, being flexible to the communication context (regarding deployed service, centralization, connection-orientation, etc) but demands a framework for collecting and controlling the path diverse information. The use of supporting records fills the gap by allowing a database for storing, inspecting, and most importantly, organizing GP information. [8] first introduced two record classes: one for maintaining the state (that is, all its attributes and characteristics) of GPs – GP Management Record (GPMR) -, and another class for holding E2E GPs information, generically reflecting paths characteristics.

In [7], this work evolved into an organized GP Resource Management Database. In the corresponding evolution, a 3-tier management framework was built, composed from a Node view, a CT view, and a GP view. While the contained resources from a Node and a GP view were already defined by the MR and GPMR, a new component was introduced for the CT level – a CT Record (CTR) -, displaying the existing GP resource view (GPMRs) contained at the corresponding CT. That way, the top-down view from the GP resource database spans from the MR to the GPMR.

These enhancements allow the establishment of a distributed hierarchical scheme, and add some new aspects. In this updated proposal, a MR points to a list of CTRs, one per each CT in which the Node CT participates, and organizes the information related to all communications within that Node CT, therefore acting as the central resource management entity of the Node CT, and the first point of access from outside components.

Each GP needs proper control, and as the element that *de facto* stores all the relevant information about it, the GPMR can achieve such a necessary feature, by having a strict relationship with the EP where the GP attaches. That is, a GPMR maps to the GP through the EP. Adding to that, such framework brings an innovative approach to nowadays networks, by providing a unified framework for controlling the communication path. Figure 2 depicts the logical disposition of records in the nodes, along with the GP architectural elements.

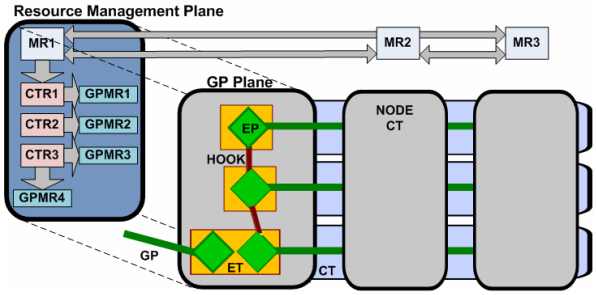

Figure 2 Records Distribution in Nodes

At the highest abstraction level, a GP is the result of the composition of lower level GPs, one for each of the technologies that the communication depends on (TCP, Ethernet, Optical Fiber, etc). In order to allow GP selection / comparison, each E2E GP is assigned with very generic attributes such as throughput and E2E delay, besides a unique identifier. In sub-GPs, the managed resources range from statistics such as throughput, SNR and end-to-end delay, to privacy and policies data, such as packet priority level and identifying keys. The nature of the stored information and the record structure depends on the class of the GP, and therefore, one GPMR class exists for each GP class. All GPMR classes derive from a Base GPMR class, which means an instantiated GPMR structure will always depend on the type of associated GP attributes Class. These Classes will be derived according to such characteristics as connection-orientation (stream vs. datagram), physical related properties (wireless vs. wired) or number of destinations (unicast vs. multicast vs. broadcast). That way, a GP will have a number of very specific GPItems that are related to the technology or service it refers to, as shown in Figure 3.

The level of management centralization is set according to the communication needs by the way these MRs are used: the basic model idea is to implement a distributed database where every node has a view of GP resources for the GPs it is involved in. On top of this model it will be possible to build any other. For some CTs, it will make sense to hold a more centralized database (e.g. Windows NT domain CT); another option is to use a more distributed database with partial views on each node (e.g. Delay Tolerant Networks or Wireless Mesh Networks). Another case may be maintaining a replicated database (e.g. unstructured P2P network), assuring data security through redundancy. A strong point in such method is avoiding the limitations of a "one-model fits all" approach.

Matters such as scalability can therefore be assured by associating to each service a specific record disposition, thus providing stability to the network and enabling presently dysfunctional mechanisms such as multipath routing or multihoming.

This papers novelty derives from a deeper description of the whole Resource Management Database, and from the first snippet demonstration of the aforementioned ideas.

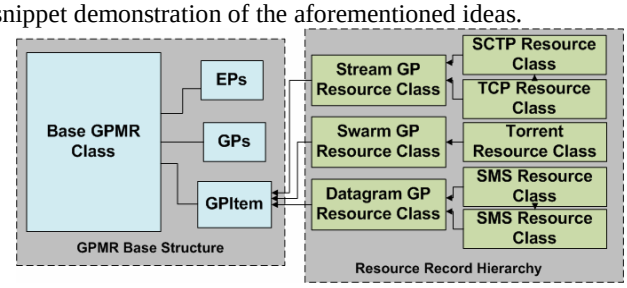

Figure 3GPMR and Resource Classes

A. Relational Model

This section details the relationship between the different main and supporting records. The whole Database is represented by the MR record, which is the SQLite file holding the other records (CTRs, GPMRs, etc). The reason for opting for SQLite was its advantageous characteristics: zero configuration ()no installation or initial configuration needed, serverless (no intermediary server process for writing/reading the database), single Database file (all data accessible through a single file), stable cross-platform database file (usable in different architectures), compactness (less than 275KB when optimized for size), manifest typing (value type is choice of user and not limited to column type), variable-length records (resulting in smaller databases), readable source code (high accessibility), SQL statements compile into virtual machine code and the source code is Public domain [9].

Each of the records is associated to an entry at the corresponding table. The set of tables existing in the MR database are as follows (refer to Figure 4):

- CTRs: lists the CTs to which the node is currently associated (CTR ID + CT name). There will usually be a limited number of CT entries (e.g. IP CT, TCP CT, LTE CT, domain CT, etc)
- GPMRs: lists the set of GPMRs (GPMR ID + CTR_ID). These exist on a relation of N:1 relatively to a same CTR. A GPMR entry cannot be associated to more than one CTR_ID, as the GPMR is the node's view of the considered GP.
- GPs: lists all GPs sessions running in the node (GP ID and GPMR ID). A same GP cannot be associated to more than one GPMR.
- EPs: lists all EPs of GPs running in the node (EP ID + GP ID);
- GP_attributes: lists all QoS or any other attribute / resource known to the node (Attribute [e.g., throughput], GP ID, Attribute_ID (determined by the associated CTR, that is, a GP existing in a specific CT has N attributes with Attribute IDs ID1, ID2, …, IDN, respectively, with those IDs being more "unique" the more specific / lower they are in the class hierarchy) and Value (may be either numeric or string, depending on the GP attribute under consideration).

Besides these main tables, one supporting table is used (represented in full light gray in Figure 4):

- Attributes_ID/CTR: this table maps attributes IDs to a corresponding CTR, in a N:1 relation. This is necessary for distinguishing the context in which each attributes is inserted, allowing for example to select all resources related to a CT, and further filter from those results. For example, one could search for all existing GP class X with attribute ID Y > 50.

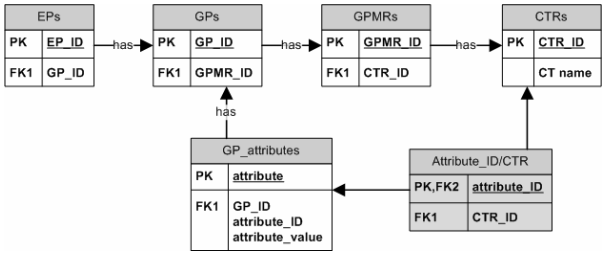

Figure 4. Resource Management Database organization

B. Utilization Example

This section presents a basic and sample example for the bootstrapping of the GP resource management database at a node. As it is an oversimplified example, the semantics should not be taken into much account (i.e., the lists and values of the properties present at the WiMAX GP).

The bootstrap of the records is as follows: i) the MR is initialized at the Node CT's creation; ii) a GPMR is initialized, being filled with the GP's characteristics; iii) if the GP is of a non-existing type to the node CT, a CTR is also initialized, otherwise the existing CTR is updated, i.e., an GPMR entry is added to the CTR.

In this example, a previously created GP object of the CT "WiMAX", with a source Entity SRC_ENT and a destination Entity DST_ENT (therefore, a unicast communication) will be added to the database. The database is initialized, and the GP is added by a add_GP command, which has as input the GP identifier (function overloading allows add_GP with multiple number of source and/or destination Entities). The code complexity hides that, by that ID, the appropriate CTR and

GPMR are updated or initialized, in case is necessary. The previous description can be seen in Figure 5. In the code snippet, a show_CTR command for displaying a list of CTRs of the current MR is used, as an example of what could be developed from the whole database. Similar functions were also written for listing GPMRs from a CTR, the GPs from a CTR, GPs that fulfill a certain requirement (e.g. RTT < 50 ms), etc, supported by the chosen SQLite database engine. By using a common API whatever the level of communication is, features such as simplified resource management or network services composition are possible.

Figure 6 shows a sample of the C++ API of the MR. These functions include CTR creation and destruction, display of a specific CTR records (i.e. CT type, associated GPMRs, etc), and additions of GP, as used in Figure 5.

Resuming, the process of records design for different communication patterns or context resumes to coding the base structure of the GPs (i.e. type of properties) based on the inherited GP attributes, while the base records manipulation methods occur taking advantage of the same initial API.

From a heterogeneous network perspective, GP-enabled devices can facilitate ad-hoc communication, with non-participant devices acting as helpers or MIH-enhanced [10] management entities, providing information from all levels, and allowing for example the selection of a path based on ANY communication level requirement (application, network, data link, …), in a pure cross-layer design.

Next section does a short description of the usefulness of resource ontologies.

```
MR* master;
GP* gp;
gp = new GP("WiMAX",SRC_ENT, DST_ENT);   // GP creation
through GP API
                                  …
master = new MR("MR.sqlite");   // database initialization
master->add_GP(GP_ID);          // filling of GPMR with GP data
master->update_GP(GP_ID, char Par, int Value)      // update of GP
parameter in GPMR
master->show_CTR("wimax");    // print of "WiMAX" CT
master->close();// Close Master Record Database
```

Figure 5 Snippet from Master Record bootstrapping

```
MR(char* filename)

void create_CTR(char* ct, int Entity)
void destroy_CTR(char* ct);
void show_CTR(char* ct)
void add_GP(GP* gp);
```

Figure 6 MR sample C++ API

C. Resource Ontologies

In [7], a section on Wireless Mesh Networks praises the need for the GP resource management database, where the information about individual channels, either static / rarely variable (e.g. frequency, queues states) or dynamic (e.g. channel utilization, average received power, cumulative physical interference) needs to be held. While this information will typically be maintained in the GP resource management database, information related to individual channels is also accessible through the GP objects through the GP API, although only locally relatively to the correspondent CT.

In order to take total advantage of the database, the definition of resource ontologies is seen as an optimal tool (also referred in the previous reference) for accurately and efficiently mirroring the cross-layer interactions. Under this GP resource ontology, the relationships between QoS attributes / resources are developed, at the backplane of the GP resource management framework, facilitating the immediate reflex of a change in a parameter in a related one (e.g. crossing RTT threshold for making Hand-over decisions). While this notion is already present in the way networks work today, the idea is to implement a unified and incremental framework of resources, simplifying resource management. The complexity of the functions of these relationships may vary, from e.g. the influence of received Power in SNR, to more mathematically advanced relationships such as activity period and channel utilization.

The real value of resource ontologies is still under research, as no real-conditions complete frameworks have been tested to date. Issues that also need to be answered are how to keep a good trade-off between the number of properties involved in the ontology to keep it efficient, which are the most important properties to select from the whole protocol stack, etc.

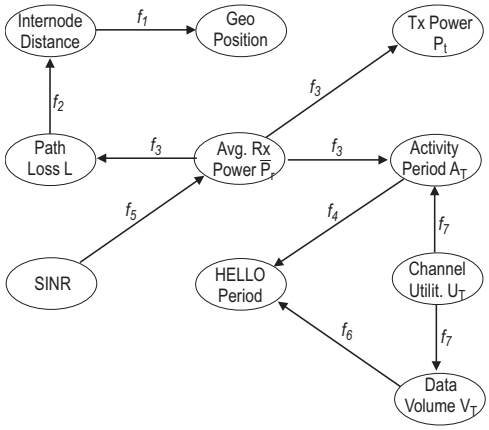

Figure 7 Example Ontology in WMNs [7]

IV. CONCLUSION

This work presented an adaptable resource management framework developed within GP architecture. Although introduced under the scope of GP architecture, an analogous solution may be developed in current networks: the clean-slate properties of the resource framework lie in its object-oriented design. The main and obvious advantage is the possibility for organizing all necessary management data through means of a restricted but simple set of functions. Thus, as opposed to traditional Internet Model, GP architecture removes the need for communication between contiguous layers, making cross-layer communications an embedded feature. Additionally, this solution should allow operators to efficiently retrieve information from any level of communication, for example for

tracking running protocols (i.e. CTs) using a single tool, something not existing today.

Another advantage is the simplicity to increment the database with new classes, resulting from the Object Oriented design features such as recursion or composition. Besides, the proposed model is inherently versatile, allowing it to be used for very distinct communication contexts, by setting up the optimal distribution / centralization balance of the information records.

Analyzing against alternative architectures, the use of an object-oriented approach adds extra value to network management, by dealing with the path as an object, allowing for example clean and effective path configuration, inspection or comparison, either inter or intra-technologically.

The improvement of network management efficiency seems as possible by means of the definition of resource ontology, a suitable concept for supporting and facilitating networking interoperability, allowing quality-aware network resource selection and composition. The development of this work is the base for providing efficient algorithms for accessing and configuring the resource management records in an efficient and reliable way, in an inherently cross-layer approach.

As interest in the concept of flat architectures [11] grows (verified in the decrease of node diversity from 3G to upcoming 4G) due to the identification of advantages such as network complexity decrease, and being a key for always-on route optimizations in mobility scenarios, the presented work may be seen as an alternative tool for simplifying network interactions, by providing a common but flexible API that absorbs most problems derived from heterogeneous networks. Thus, our solution removes management centralization, network levels and dependencies that forbid other architectures from obtaining desired scalability.

Upcoming work consists on the definition of necessary signaling, and further integrating the work developed in SQLite with the Future Internet Toolbox (FIT) [12] for a preliminary evaluation.

ACKNOWLEDGMENTS

This work has been carried out in the framework of the IST 7th Framework Programme Integrated Project 4WARD, which is partially funded by the Commission of the European Union. The authors would like to thank his colleagues in the 4WARD Work Package 5, aptly titled Generic Paths, for fruitful discussions.

The views expressed in this paper are solely those of the authors and do not necessarily represent the views of their employers, the 4WARD project, or the Commission of the European Union.

REFERENCES

[1] Roberts, J., “The Clean-Slate Approach to Future Internet Design: A Survey of Research Activities”, Annals of Telecommunications, Springer- 2009

[2] GENI Project: http://www.geni.net

[3] A Recursive Network Architecture, http://www.isi.edu/touch/pubs/isi-tr-2006-626/

[4] “The SILO Architecture for Services Integration, controL, and Optimization for the Future Internet”, ICC 2007

[5] Paul, S.,Pan, J., Jain, R., “Architectures for the Future Networks and the Next Generation Internet: A Survey”, 2009

[6] The FP7 4WARD Project, http://www.4ward-project.eu/

[7] Randriamasy, S. et al., “Mechanisms for Generic Paths”, 4WARD deliverable D-5.2, December 2009

[8] Figueiredo, S., Lourenço, J., Aguiar, R. L., and Neto, A., “Taxonomy for GP-aware mobility”. Proc. of the First International ICST Conference on Mobile Networks and Management, 2009

[9] http://www.sqlite.org/different.html

[10] IEEE 802.21 Standard, “Local and Metropolitan Area Networks – Part 21: Media Independent Handover Services”, January 2009.

[11] K. Daoud, P. Herbelin, N. Crespi: K. Daoud, P. Herbelin, N. Crespi, “UFA: Ultra Flat Architecture for high bitrate services in mobile networks”, In IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008 (2008), pp. 1-6.

[12] C. Dannewitz, T. Biermann, M. Dräxler, F. Beister, and H. Karl, "Prototyping with the Future Internet Toolbox", Proc. 6th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom), May 2010