Proposal of a Clean Slate Network Architecture for Ubiquitous Services Provisioning

X. Sanchez-Loro, J. Casademont, J. Paradells
Wireless Networks Group – Telematics Department
Technical University of Catalonia
Barcelona, Spain
{xsanchez, jordi.casademont, teljpa}@entel.upc.edu

J. L. Ferrer, A. Vidal
I2Cat Foundation
Barcelona, Spain
{jlferrer, albert.vidal}@i2cat.net

***Abstract*— The Pervasive Computing field is almost always addressed from application, middleware, sensing or Human Computer Interaction perspective. Thus, solutions are usually designed at application level or involve developing new hardware. Although current layered network architectures (mainly TCP/IP stack) have enabled internetworking of lots of different devices and services, they are neither well-suited nor optimized for pervasive computing applications. Hence, we firmly believe that we should have an underlying network architecture providing the flexible, context-aware and adaptable communication infrastructure required to ease the development of ubiquitous services and applications. Herein, we propose a clean slate network architecture to deploy ubiquitous services in a Pervasive and Ubiquitous Computing environment. The architecture is designed to avoid hierarchical layering, so we propose a service-oriented approach for a flow-oriented context-aware network architecture where communications are composed on the fly (using reusable components) according to the needs and requirements of the consumed service.**

Keywords- Future Internet, Pervasive and Ubiquitous Computing, Profiling, Context-Awareness

I. INTRODUCTION

Current network architecture design is based on hierarchical layered models like OSI and TCP/IP stacks. In these models, networking functions and protocols are grouped in layers, according to a common objective and scope. Thus, each layer performs different networking tasks, restricting inter-layer communication to immediately adjacent layers. In theory, each layer is in charge of a group of functions, but in practice functions overlap at different layers, adding protocol overhead and blurring the layered structure of the protocol stacks.

This rigid layering approach derives in monolithic network architectures, lacking flexibility. First, network functions are executed regardless of the characteristics of the surrounding context, underlying network technologies and capabilities of the devices involved in a communication. So, for some situations, the same functions can be redundantly executed at different levels, degrading communication performance and wasting computing resources. Even worse, in certain environments, the execution of certain functions can be counterproductive for the correct operation of an application or network service (e.g. TCP's congestion control in wireless networks), obliging to modify existing protocols to adapt them to environments with restrictions [6].

Second, inter-layer communication is strongly restricted. This has led to different cross-layering approaches to enhance protocol and application performance in wireless environments where traditional protocols show a poor performance [5]. These cross-layering solutions, violating the layered structure of the stack, further complicate the situation, bending the model and its related standards. These practices pose serious interoperability issues resulting in an increased complexity of the network architecture.

Third, another cause of strife is the existence of new sub-layers like MPLS at layer 2.5, IPsec at layer 3.5, and TLS at layer 4.5. These sub-layers are features not considered in the original design of the stack and they have been implemented as patches as a consequence of the lack of flexibility of the TCP/IP stack.

Fourth, middle-boxes (NATs, proxies, gateways, etc) erode the end-to-end model as new features and participants are placed in-network without control and knowledge of the edges. These now common solutions further complicate the situation as they were not considered during the original design and development of the Internet (they did not exist then) [9]. Similar situations will surely arise during the following years as technology and applications evolve, and the TCP/IP stack lacks flexibility to clearly and easily deal with them.

Most of these issues derive from the fact that TCP/IP stack was designed with wired networks and mainframes in mind [3, 9]. The first users of these earlier networks were a trustful community of people with high technical skills, enforcing the end-to-end arguments [2, 9]. Protocols were tailored for the capabilities of the nodes and technologies of the time, whilst the applications were much simpler than today's, in fact most of them were simply replicas of the services of an operating system on a network (telnet, FTP and RJE). In the following decades, Internet applications and network services evolved, increasing its complexity and requirements, diverting them from a strict end-to-end philosophy. Quality of service (QoS) and security were introduced as critical issues. With the rising of wireless networks (mobile, ad hoc, mesh, sensor) and mobile devices, new applications and issues arise: mobility, hostile and time variant access media (interferences, lower bandwidth, higher error rates, higher latency, etc.), intermittent connections, energy restrictions, multi-modality issues, device capabilities issues, localization, nomadism, roaming, context-awareness, network and device heterogeneity, transcoding, etc.

This work was supported in part by the i2CAT Foundation and the Spanish Government through the MECD and FEDER project TIC2006-04504.

978-1-4244-5160-9/09/$26.00 ©2009 IEEE

Now, there's a myriad of different devices, applications and network technologies. These advances have allowed new computing paradigms like Pervasive and Ubiquitous Computing or the Internet of the Things, to name a few, but these paradigms shift from strict end-to-end arguments and pose very different requirements and philosophy than original TCP/IP applications -some don't even use the IP protocol. They require new modes of interaction between nodes and components of network services not fulfilled by current network architectures. So, network services should evolve on an architectural framework to become flexible, ubiquitous, composable, dependable, secure, context-aware and adaptable in execution time. Following this trend some discussion about "clean slate" re-design of the Internet [10-13, 8] has arose during the last years and, we believe it will intensify in the next years. Now, it is time to design the Network with the characteristics we would wish to take for granted in 10 years.

II. DESIGNING A NEW NETWORK ARCHITECTURE

First, we should bear in mind when designing a new network architecture[1] that it should be flexible and adaptable enough to perform reasonably well in all kind of environments, without making assumptions about execution environment, infrastructure support or a minimum set of device capabilities. It should be truly ubiquitous, providing tools for consuming network services anytime, anywhere, and anyhow (that is with any device, any platform); thus, integrating all kind of edge networks, platforms and devices [4]. This is very important because most advances in the networking field come from the edges. Therefore, if we can design a flexible architecture, adaptable to context variations, that works with minimalistic devices in restricted networks like Wireless Sensor Networks; it should be easier to extrapolate it to work in more complex environments, without capacity restrictions, like wired computer networks. Hence, we would like to invert the tendency to design complex protocols for wired and backbone networks (with computers and routers as reference nodes) and then adapting them to environments with restrictions, approach that poses a lot of implementation, design and performance issues. Furthermore, restricted devices like sensors/actuators and objects are becoming a real majority in the Network (and they will keep growing in numbers), thus we cannot be oblivious of their impact on networking applications and design a model that accommodates them.

We need a shift from strict end-to-end arguments, building a network architecture that provides more intelligence to the network-side whilst still leaving decision-making processes to the end-points [1-2]. In our opinion such architecture should have the following characteristics and features:

- **Context-awareness and dynamic adaptability during execution time**. It must take into account the capabilities of the nodes, services and network links to establish new routes and to manage existing ones. So, mechanisms to interchange context information between entities in ad hoc and structured ways must be supported.
- **Oriented to service/resource interconnection and not machine/interface interconnection**. We propose to shift the focus on network addresses to a service/data-centric approach that allows the semantic discovery of services and resources (including data objects), easing the interaction with network facilities.
- **Semantic identification and addressing of nodes, resources and services**[2]. Service discovery and, hence, routing must be based on the semantic description of the desired service, including security functions. This way, we avoid making explicit addressing (and naming) mandatory. Besides, existing addresses (locators and identifiers) are treated as another characteristic of the service/node/resource. Furthermore, when used, addressing schemes should be designed to be dependent on the location of entities in a network, but route independent [3]. This semantic context-aware service-derived route discovery approach provides intrinsic support for mobility, multihoming and nomadism.
- **QoS integrated into routing and service discovery**. Discovery, establishment and management of routes must be based on requested QoS and resource availability.
- **Flexible design and execution**. Network functions must be allocated according to each situation and not in a monolithic way. Thus, functions must be allocated all along the route, executing just the desired functions at each hop, section of hops and end-to-end and applying them just to the desired transmission unit (symbol, frame, packet, etc.). Flexible support for different semantic schemes or vocabularies for identifying services, resources and nodes and to describe their capabilities must be also devised. This flexible support must be also extended to different schemes for specifying desired/requested and provided QoS.
- **User empowerment in service choice and routing**. In our opinion, with the diversification and popularity of the Internet and the maliciousness of some of their players [2], users should be provided with mechanisms that allow them more control over their communications. This control should be reflected in flexible routing and service selection. So, a service requester must be able to choose from matching service responders which specific service wants to consume. Also, if desired, source must be able to specify preferred and trusted carriers/domains and blacklist distrusted or malicious domains, this way end-points

[1] By architecture we mean: "a set of rules and constraints that characterize a particular style of construction" [3]. So, applied to the context of network design, it defines the network design model; that is the global technical principles of design of the network.

[2] Users and resources are discovered through the semantic discovery of the service(s) that provides access to them, acting as their interface with the network (object delivery services, user communication services, etc.)

have a certain degree of control over which routes their communications follow

- **Security functions must be fully integrated in its design**. In our opinion, security must not be an addendum. Hence, service discovery/consumption takes into account available security features. Other points to assure are: data integrity and confidentiality, plus user privacy and confidentiality. Also, we believe that another important characteristic is traceability, i.e., finding the path to the traffic source for a specific communication. As each connection is established for service consumption, traceability can be achieved by univocally tagging each single traffic flow. Therefore, we make use of a unique tag (Session Identifier) to identify and route each traffic flow.

III. PROPOSAL OF A NEW NETWORK ARCHITECTURE

Herein, we propose a new network architecture to deploy ubiquitous services in a "clean slate" environment. The architecture is designed to avoid hierarchical layering, so we propose a service-oriented approach for a flow-oriented context-aware network architecture where communications are composed in situ (using reusable components) according to the needs and requirements of the consumed service. Thus, we define the network as a set of nodes and services. Nodes are physical (device) or virtual entities (cluster) possessing networking capabilities able to consume and to provide services to itself and other entities and services. They provide the necessary environment for service composition and execution.

Services are classified into atomic and composed services. Atomic services are those individual functions[3] commonly used in networking protocols (i.e. acknowledgments, sequence numbers, flow control, etc). These are well-defined and self-contained functions, used exclusively to establish communications for consuming composed services. Composed services are network applications with a wider scope than just establishing communications (e.g. printer service, directory service, file transfer, instant messaging, presence, etc.). Each composed service or application imply consuming different atomic and, sometimes, other composed services; appearing possible dependences between them. Also, they can involve one or more nodes, depending on the complexity of the service.

In this model, in order to obtain the desired behavior, functionality and QoS constrains, communications are established concatenating atomic services into a workflow for consuming a certain composed service. So, atomic services are the building blocks used to establish communications and to deliver data in a self-adaptable, self-configurable and context-aware way. They are allocated amongst involved nodes, as required by conditions of temporal context and service requirements. In this way, all functions are used only when and where they are required, so that we assure that there is no function overlapping or usage of counterproductive functions.

[3] A function is a set of self-contained and well-defined instructions executed with a common purpose in order to provide a certain logic mechanism for data-interchange between services

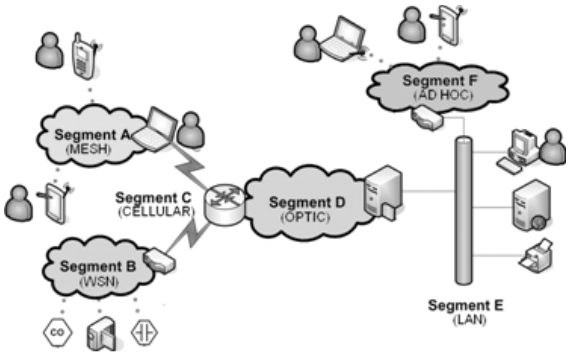

Fig. 1. Example of heterogenic networks

For instance, there can be heterogenic networks where a node may require very different network functions to communicate with different nodes in order to consume the same or a different composed service (see figure 1). On one hand, there may be network segments with reliable communication environments neither requiring error correction nor error recovery, or perhaps very basic functions like a small CRC computation. This could be the case for nodes connected with wired and reliable links (e.g. segment E in figure 1). On the other hand, other network segments could require strong error detection and recovery mechanisms in order to accomplish reliable data delivery in front of high error rates in unreliable links (e.g. segment A, B, F). Function allocation not only depends on network and link state but also on device capabilities. In this way, there may be nodes with strict restrictions in battery, memory and CPU preventing them to perform intensive operations like message sequencing, data transcoding or executing complex timed state-machines (e.g. segment B).

Hence, in order to provide and consume (composed) networked services in a truly ubiquitous fashion (according to desired functionality, behavior and QoS constraints), atomic services must be suitably allocated along the communication path where they are actually needed and tuned/configured accordingly to accomplish the application requirements. Thus, atomic services can be executed in a per-hop, per-section (between two non-adjacent nodes) and/or end-to-end basis (section ranging the entire route).

A. Atomic Services

Each atomic service (see figure 2) provides one concrete and well-defined networking function (along with the reverse function, if any). Different algorithms and implementations of an atomic service could exist (i.e. different congestion control algorithms), and co-exist in the same node, using attributes to both describe the different possibilities and to tune/configure the atomic service in order to use it to fulfill specific workflows needs.

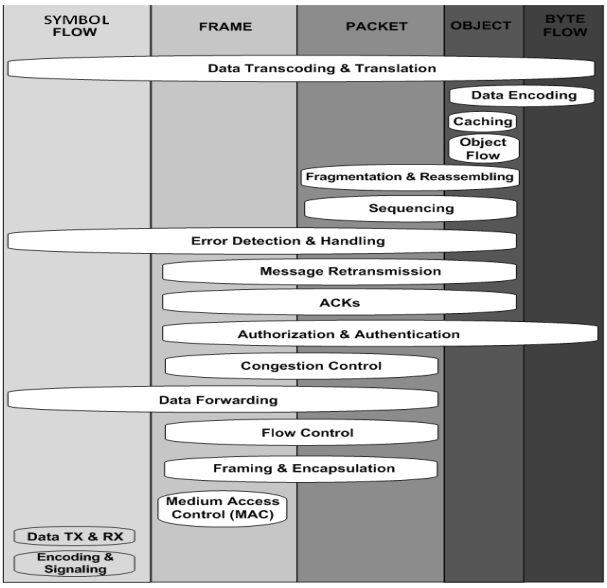

Fig. 2. Atomic services

1) Service granularity

Atomic services can be executed with different levels of granularity, depending on desired functionality. For each level, a different unit of information is processed/affected. We define 5 levels of granularity for atomic services:

1. Symbol flow: At symbol flow level, atomic services are executed affecting the whole communication flow at symbol level paying no attention to logical abstractions, akin to circuit-oriented communications[4].

2. Frame: At frame level, services are executed on per-frame basis. So, all transformations, headers and responses use physical frames as basic I/O unit and interfaces. The frame level has its own sublevels of granularity:

a. Default: service execution affects the entire frame.

b. Payload: service execution only affects frame payload.

c. Overhead: service execution only affects frame overhead.

3. Packet: At packet level, logical packets are the basic I/O unit. Identically to the frame level, packet level granularity has the default, payload and overhead granularity sublevels.

4. Object: In object level, services are executed on a per-object basis. An object is an encapsulated/structured data resource and may contain further objects inside. In order to feasibly implement this level of granularity, support for the object flow atomic service must be provided in all the nodes executing services with object-level granularity.

5. Byte flow: In byte flow level, services are executed affecting all the session's data flow (without overhead) and treating it as a raw stream of bytes.

[4] In this case, services are negotiated as usual (see section IV.D), agreeing on a *Encoding & Signaling* service implementation that allows working on circuit mode (allocating a dedicated frequency, TDM scheme, etc); giving support for a continuous traffic flow and just tagging the traffic flow at the start of the session with a session id

B. Semantic Service Identification

Ubiquitous computing relies on the idea that users should be able to consume network services anytime, anywhere and anyhow (with any device and platform). This implies mechanisms to create, discover, negotiate and consume composed services in a flexible and context-aware way. Service consumers (users and other entities) may not know which device provides the desired composed service, they may even not know the name or the identifier/locator of the service or its provider; but they do know the characteristics of the service they want to consume. Thus, consumers must be able to describe the desired capabilities of the requested service and the network must be able to resolve if there is any service matching this description. In this way, service consumers describe the desired service using semantic constructions like "I want a color printer close to building X with toner and paper". Each node knows its own capabilities and which composed services it provides and their characteristics, which are described in node and service profile instances. Therefore they can match against the attributes of their service profiles if any of their composed services complies with the desired functionality. This process is called semantic service identification.

In order to be feasibly implemented, network nodes must share a common knowledge base or ontology; that is a common attribute semantics and syntax. Although different ontologies may be supported, all nodes must support the minimum identification ontology. This basic ontology is designed to be minimalistic, in order to be supported by all kind of devices and platforms with enough ease (in terms of memory, computing power and energy), but still providing enough level of expressiveness and completeness when building semantic constructions. Hence, the nature of the relations expressed by each attribute (i.e., "is a", "has a", "belongs to", etc.) is inferred from its own definition. Attributes are defined for describing node capabilities (CPU, memory, network interfaces, battery, etc.), temporal context characteristics (location, domain), atomic services characteristics (type, supported granularity, dependences, configuration parameters, etc.) and composed services characteristics (I/O behavior, negotiation scheme, description, provider, etc.) Besides, in order to minimize the amount of information transferred, attribute syntax is dictionary-based.

The following operators can be applied to attributes when constructing semantic descriptions:

- Logical operators: AND, OR, NOT
- Operators <,>, = for comparing attribute values
- Regular expressions

IV. Proposal of a Protocol for Negotiating and Consuming Ubiquitous Services

In order to allow communication creation and atomic services allocation, we have designed a protocol based on ad-hoc routing protocols for ubiquitous discovery, negotiation and establishment of communications between service creators and service consumers. This protocol is devised for pure ad hoc environments, with no infrastructure support and dynamic

topology; so we use a controlled flooding scheme for disseminating the communication requests. We choose flooding as it does not require any extra information for its operation. However, other ad hoc/WSN routing schemes [7] like gossip algorithms, Directed Diffusion and Low-Energy Adaptive Clustering Hierarchy (LEACH) may be used as well with ease.

In case of fixed and structured networks, there is no need for ad hoc/topology-changing schemes, therefore current standard techniques based on routing tables with neighboring nodes/networks should be used.

A. Protocol Characteristics

The protocol is designed to discover network services, and thus routes to reach them, on-demand searching for all feasible composed services; those that meet the requested QoS (and functionality). We should note that we use the term QoS in its widest sense, including traditional QoS metrics like bandwidth, delay and error rates (*flow QoS*[5] in the text), but also the atomic services required to obtain the desired communication behavior and functionality, not just performance (*functional QoS* in the text). This includes security functions and other functions like transcoding and caching. Therefore, QoS and security are integrated in service discovery and negotiation. Service paths (i.e., routes to matching services) are discovered and negotiated according to composed service profile and characteristics of involved links and nodes.

Resources are reserved during service path negotiation, emulating a virtual circuit. We define the service communication paths as unidirectional, in order to allow a different configuration based on each link direction. However, for the returning control traffic of some atomic services (i.e., ACKs, etc.) a reverse path will be created during service negotiation.

Services and nodes are identified semantically through attributes; avoiding using just addresses to identify a node's interfaces -addresses receive same treatment as the rest of attributes. Hence, arbitrary interface addressing is substituted by semantic description of the desired service. According to this description, requests are routed searching for its final destination: a service entry-point able to provide the requested service. When a node provides or participates of a service that matches this description, it responds. Consequently, macro-mobility (inter-domain) and nomadism are supported by this semantic routing scheme. However, mechanisms for route management and reallocation are needed to cope with route degradation during micro-mobility (intra-domain). Also, multihoming is supported when required (addresses are just attributes), but more important, multihoming can be dodged by semantic identification as we search for a service or node, but not for a certain network interface address.

Profiles describing the capabilities of the nodes and its related services are interchanged between neighboring nodes in Hello messages. So, each node stores a profile database including a summary of the capabilities of the nodes and services accessible from each network interface in order to control flooding during service discovery. These Hello messages are also used to monitor link state. Besides, there is also the possibility of relying on a context information provisioning service, in case of available infrastructure support.

B. Negotiation schemes

We define three negotiation schemes in the protocol: 3-way handshake, 4-way handshake and no negotiation according to different communication schemes.

The *3-way handshake* is the main scheme of negotiation (see figure 3). It consists in a Communication Request message (CReq, see figure 4) describing the desired service functionalities and QoS requirements. QoS parameter values are divided into minimum and, optionally, optimum values. They specify desired metric values (flow QoS) and network functionalities (functional QoS). Furthermore, the CReq is tagged with a session id field to univocally distinguish that flow.

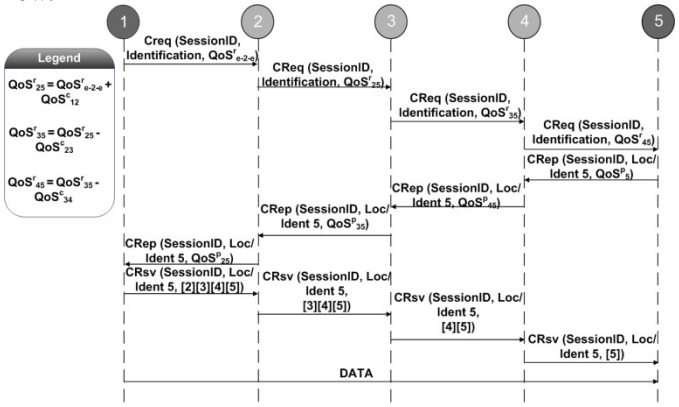

Fig. 3. Service Negotiation Flux Diagram

Intermediate nodes forward CReq using a controlled flooding scheme, until it reaches a matching service. Forwarding is constricted by semantic identification, timer expiration and feasibility of meeting QoS requirements (both in parameters and available functionalities). As in MANET/ad-hoc routing protocols, a basic Expanding Ring Search technique is also used in order to restrict range of CReq flooding. Each intermediate node modifies flow QoS requirements to reflect the consumed resources at each hop (see figure 3). This flow QoS recalculation can be achieved as we have included synchronization information in periodic signaling protocol messages. This way, using classical HELLO messages, information can be properly distributed. After forwarding a CReq, each node creates a reverse route to forward Communication Response messages (CRep, see figure 4) to source node. In order to facilitate routing and session management, the destination includes a valid locator (by default) or a univocal identifier in the reply (when source explicitly requested it).

Each node with a service profile matching the service requirements replies (unicast) to each arriving CReq with a CRep; reporting the QoS parameter values that could be provided (see figure 3). Intermediate nodes complement these values of QoS as CRep are forwarded. In this way, they

[5] It must be pointed that the data forwarding atomic service is responsible of forwarding data messages to its next destination. Thus, it is in charge of scheduling forwarded messages, dictating the PHB (Per-Hop Behavior) needed to comply/match with QoS agreements.

indicate functions that they are able to perform. They also report the aggregate metrics that the route can assure. For some parameters, this information is not enough to decide where functions should be allocated. Thus, some parameters need to be disaggregated, stating actual values at each link or node. These include node capabilities like CPU, memory (buffering), battery and link characteristics like error probability and available bandwidth.

After receiving various CReps, CReq originator decides the service path that will be used to establish the service session, sending a unicast Communication Reservation (CRsv) message indicating which functions (hop and segment) must perform each node in the service path. As CRsv message is forwarded, each intermediate node erases its own orders, concealing them from following nodes. At the end, the destination node receives its instructions for end-to-end functions. At this point, service session is established and data can be transferred (see figure 3). Data forwarding is based on Session Identifier look-up to determine next hop, output interface and functions to be executed. Scheduling of the data transfer to achieve required QoS metrics is managed by the data forwarding atomic service.

It must be pointed that session release can be explicit or implicit (timer expiration). In the explicit way, source indicates the session identifier of the session to be released and resources are orderly freed along the route. Explicit release is recommended since resources are freed when the session actually ends whilst in implicit release resources are wasted waiting for timer expiration. There is a lifetime field in the CRep packet that indicates the inactivity period supported before the route is deemed to be expired. This value is set by the originator of the reply and may be decreased (up to a minimum threshold) by intermediate nodes according to their resource and energy saving policies.

The *4-way handshake* scheme is used to negotiate additional parameters of the allocated functions. It consists in a three-way handshake adding a way to agree the parameters of usage of certain allocated functions (e.g., cryptographic keys, negotiating cipher algorithms, transcoding formats). This scheme is mainly devised for secure services, as security functions usually need this additional negotiation for their configuration.

The *No negotiation* scheme is designed for sending small data (about 1 data packet.) during an instant session. As its name suggests, there is no session negotiation and data travels through the network embedded in discovery messages. Thus, data will flood the network unless there is an existing route to the destination; in this case the communication is unicast. This scheme makes use of the following atomic services: data transmission and reception, framing, MAC, identification and data forwarding. This scheme is devised for simple sensing applications, where nodes send small amounts of captured data at spaced intervals and, therefore, there is no need of wasting resources establishing and maintaining a session.

V. CONCLUSION

Future Internet design should address most of the shortcomings of current Internet architecture. Specially, we believe it should address the lack of ubiquity, pervasivity and context-awareness in the TCP/IP stack. This focus on ubiquity implies designing an architecture that suits the requirements of sensor/actuators and object networks. WSN and other edge networks are becoming a majority in the Internet and they require solutions that fit their needs and restricted capabilities. Furthermore, it is easier to implement innovations in the edges than in the core of the network. So, herein, we have presented the foundation of a clean slate network architecture that focuses on service ubiquity, adaptability and context-awareness in restricted edge networks, like WSNs. From a Pervasive and Ubiquitous computing view, this architecture solves/mitigates some of the issues and shortcomings of the current Internet architecture providing features like: context-awareness and dynamic adaptability during execution time; flexible allocation and execution of network functions; traceability; QoS, security and service discovery supported by the core of the architecture; routing based on the semantic description of services (non-mandatory use of addresses, support for mobility, nomadism and multihoming); QoS and resource availability integrated into routing and service discovery; enhanced user control; etc.

Although we can devise solutions that work well on paper, we need to prove their feasibility. So, now, we are working on a test-implementation to prove it and to test its performance compared to current solutions. We know it is difficult to surpass the performance of current solutions as they have been polished for more than 20 years, but we expect to at least match it in a first attempt, whilst providing further functionality and flexibility.

Some future work on the architecture includes addressing different issues: mechanisms to address communications and service discovery across domains in a scalable way; exploring optimal strategies for atomic service allocation; solving intra-domain mobility and route degradation; further exploration on syntax, rules and semantics of the different vocabularies for QoS, resource and service description; exploring strategies for enhancing flooding control during service discovery; exploring suitable semantic addressing schemes for assigning locators and identifiers; etc.

REFERENCES

[1] NewArch Project: Future-Generation Internet Architecture. http://www.isi.edu/newarch/

[2] D. Clark, J. Wroclawski,, K. Sollins and R. Braden, "Tussle in Cyberspace: Defining Tomorrow's Internet". ACM SIGCOMM 2002, August 2002.

[3] J. Day. “Patterns in Network Architecture: A Return to Fundamentals”. Prentice Hall PTR (January 6, 2008).

[4] D. Clark, C. Partridge, R. T. Braden, B. Davie, S. Floyd, Van Jacobson, D. Katabi, G. Minshall, K.K. Ramakrishnan, T. Roscoe, I. Stoica, J. Wroclawski and L. Zhang. “Making the World (of Communications) a Different Place”. ACM SIGCOMM Computer Communication Review. Volume 35, Number 2, July 2005

[5] VineetSrivastava and MehulMotani. “Cross-Layer Design: A Survey and the Road Ahead”. Communications Magazine, IEEE, Vol. 43, No. 12. December 2005

[6] X.Sanchez-Loro, Victoria Beltran, JordiCasademont and Marisa Catalan. “Ubiquitous Web Access: Collaborative Optimization and Dynamic Content Negotiation”. The 2nd International Workshop on Interactive Multimedia & Intelligent Services in MUC. Busan, Korea, 24-26 April 2008

[7] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Elsevier Ad Hoc Network Journal*, vol. 3, pp. 325-349, 2005.

[8] D. Stevenson, R. Dutta, G. Rouskas, D. Reeves and I. Baldine. “On the Suitability of Composable Services for the Assurable Future” Milcom '07 October 2007, Orland FL

[9] D. D. Clark. The design philosophy of the DARPA internet protocols. ACM SIGCOMM, pages 106–114, Aug. 1988.

[10] GENI.net Global Environment for Network Innovations. http://www.geni.net/

[11] NSF NeTS FIND Initiative.. http://www.nets-find.net/index.php

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

[13] "AKARI" Architecture Design Project for New Generation Network”. http://akari-project.nict.go.jp/eng/index2.htm