Ambient Networks Integrated Prototype Design and Implementation[1]

Cs. Simon, R. Rembarz, P. Pääkkönen, H. Perkuhn, C. Bento, N.Akhtar, R. Agüero, T. Katona, P. Kersch

csaba.simon@tmit.bme.hu, rene.rembarz@ericsson.com, pekka.paakkonen@vtt.fi, heiko.perkuhn@ericsson.com, cbento@criticalsoftware.com, n.akhtar@eim.surrey.ac.uk, ramon@tlmat.unican.es, tkatona@hit.bme.hu, kersch@tmit.bme.hu

***Abstract*—This paper presents the design and realization process of the Ambient Networks Integrated Prototype. The approach of developing a prototype that integrates various components developed by different teams working in parallel is a cornerstone of the project's validation strategy. We offer other projects a case study that helps speed up their design and development process for complex research prototypes. We present the top-to-bottom approach starting from usage scenarios and leading to the list of software modules to be developed. The structure of the prototype is explained, with focus on registration, lookup, naming and security framework. By this we provide an insight into the internal structure of a future global Fixed-Mobile Converged (FMC) network.**

***Index Terms*—Ambient Networks, prototype, FMC**

I. Introduction

The ever-decreasing size and ever-increasing performance of computational and communication devices opens the way towards the so-called Ambient Networking vision. According to this concept, in the near future, an average person will be surrounded by dozens of devices – by laptops, PDAs, smartphones and several supporting devices still to come, which will make life easier (e.g., medical sensors). The World Wireless Initiative – Ambient Networks project [1] funded by the European Union is targeted to research various networking-related aspects of telecommunication systems of the future. Technologically speaking, this translates into the design of an affordable and scalable mobile communication network. It also aims at creating mechanisms that enable efficient use of resources in an environment populated by a multitude of devices, technologies and business actors.

A major challenge addressed by the project is the harmonization and interworking of the potentially large number of networking technologies that will be available. In order to address these challenges the project adopted the design paradigm of horizontally structured mobile systems that offer common control functions to a wide range of different applications and air interface technologies. It is envisaged that each Ambient Network (AN) has its own set of control function and different ANs dynamically join and separate for diverse reasons.

Current networks have an increasing divergence in the network control layer: different control environments are established to facilitate services like VPNs, security, mobility, QoS, NAT, multicast, etc. Thus the control of such services is becoming increasingly fragmented. The Ambient Network approach alleviates this problem by creating a domain-structured, edge-to-edge view of the network control [6]. A new architecture element that implements the required control functions, called Ambient Control Space (ACS), has been introduced [2].

This paper describes the design process for implementation of such a complex ACS and documents the experiences accumulated during the prototype implementation for proof-of-concept. The integrated prototype presented here can be used to validate and demonstrate the internal functionalities as well as the underlying concepts of the ACS. The EU, through its string of Framework Programmes [3], has supported several successful projects, where different concepts in Beyond 3G and FMC networks have been prototyped. Nevertheless, the Integrated Projects in EU IST FP6 call concentrated much larger research resources, targeting wider aspects of a certain research area. Broader scope and increased resources resulted in a design and implementation challenge. While earlier demonstration activities in the AN project, similar to other EU IST projects, have focused on a few or even only one concept per demonstration [4] [5], the prototyping work described here integrated all the important concepts proposed by the project. The differences in this case are not only the issue of coordinating the work of a group of implementers an order of magnitude larger than usual, but the identification of bottlenecks and the design of a platform that glues all the modules implementing specific features.

At one hand, there are well-known solutions used for large project developments, but the project lacks the resources and time to apply a heavy-weight design phase. Moreover, the structure of the project resembles a web of autonomous partners (peers), similar to open software developers groups, rather than a strict hierarchy of the research division of a company. On the other hand there was a strict deadline to meet and clear objectives to reach. As a conclusion, the challenge was to have an enterprise level strict implementation path without the means available in a strict enterprise environment.

[1] This work has been performed within the Ambient Networks project, supported by the European Union's IST FP6.

The first answer to this challenge was to identify, through a top-to-bottom approach, the chief modules to implement and the relations among them. This process involved lot of effort from the partners as this is a peer-to-peer fashioned decision process; it replaced the hierarchical decision that sets priorities in a business-driven research company. We detail these aspects in Section II. Secondly, since there were quite a few modules to integrate, it was not enough anymore to let the implementers of a module to develop their supporting libraries and mechanisms (e.g., the messaging, presence, notification and lookup services, to name just a few). In the first phase of the project this path was followed, but while integrating them, it turned out that even those solutions that are based on the same theoretical concepts, implementation differences were practically not feasible to merge. Therefore, a selection phase was introduced, where a requirement list was put forward and then the closest match to the ideal solution was chosen.

The rest of this paper is organised as follows. Various aspects of the scenarios used to drive the prototype work are discussed in Section II. Section III describes the design process for the prototype, while Section IV presents the realization of the AN key features.

II. Demo Scenarios

A. Joint Use Case

Initially in the project a common storyline i.e., Joint Use Case (JUC), was created to serve multiple purposes. Firstly, it provides the parties working on different parts of the ACS with a common goal (integrated prototype) they can work towards. On a conceptual level, the scope of the developed functionality is usually very large. The storyline helped to narrow down the potentially broad range of functionality and focus on the important, project-level aspects. With a clear picture of what functionality is needed, this process of selection and prioritization was greatly facilitated. Secondly, working along the lines of a concrete application scenario ensured a clear focus on the usefulness of the concepts for the different actors, e.g. for mobile users or network operators. It gave concept developer teams the opportunity to test and validate their developments against a concrete use case and also examine interactions with other teams. Finally, the experiences learnt during the integration of the innovative AN concepts into one prototype are fed back to concept developers, refining the overall AN specification.

The JUC is comprised of seven scenes. As a framework, the setting of a highly mobile business traveler, who travels to a business meeting, has been selected. This involves interactions with stationary environments (home, train station) as well as moving environments like a car or a train. The JUC and the business aspects related to it have been described in detail in [4].

B. ACS mapping

In order to evaluate and validate key concepts of the ACS, a correlation of the JUC with the ACS functions had to be established, i.e. the scenes of the JUC were mapped to the corresponding ACS functionalities (or Functional Entities) used therein. Thus the key concepts to be validated through the project boiled down to interactions inside and/or among a set of Functional Entities (FEs). The concept of a Computational Process (CP) has been defined, for abstracting and better understanding such interactions inside a FE or between FEs.

The process of mapping the JUC into CPs is called *ACS mapping*. Figure 1 illustrates the ACS mapping and the way it is used to create demonstrations from the JUC. Initially, a project-wide JUC was developed that collects all concepts from the AN project into a common storyline. Then, the JUC was mapped into CPs, focusing on the interaction between FEs. Finally a set of dedicated, standalone, demonstration scenarios were created.

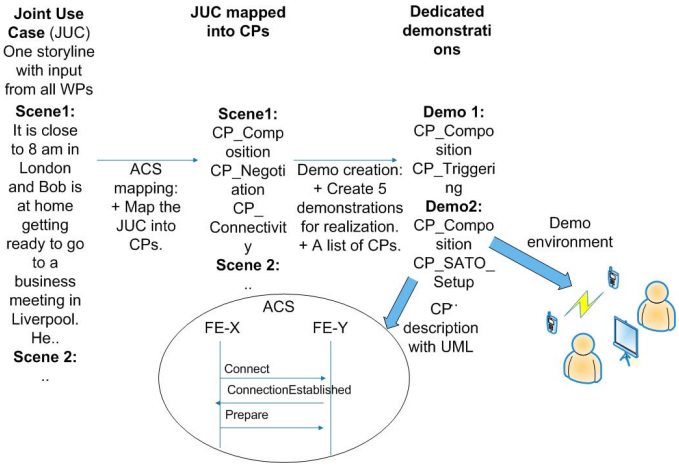

Figure 1: Demonstrations to be realized from the JUC

For the realization of the demonstrations, the common storyline was revised from the implementation point of view, and the practical environment setup (terminals, access technologies, applications etc.) described. Additionally, the list of important CPs required for the implementation of the demonstrations was identified. This mapping helped identify the set of key AN concepts to be demonstrated, which included: *Plug and Play ACS*, the ability to add/delete functionality as and when required; *Security domains and policy framework*, to allow secure interaction between AN entities; *Composition and composition management*, which deals with dynamic agreements between Ambient Networks; *Node ID management,* a novel internetworking architecture, based on node identifiers; *Service Aware Transport Overlays* (Overlay management); *Access selection, which* manages the heterogeneous wireless accesses on an efficient way. The complete list of concepts and their in-depth description go beyond the scope of this paper, but a more detailed description can be found in [4].

III. Designing An Integrated Prototype

A. Requirements for the design

In the previous section, we described the process that led to the subset of FEs to be implemented. This section describes

the way the ACS itself, which is the glue or a platform that assures the interaction of the FEs, was designed.

The requirements for design of the integrated prototype can be divided into two areas: Firstly, coherence with architectural principles [2], and secondly, practical feasibility.

One of the cornerstones of the AN architecture is the concept of an extensible control space, the ACS, that allows for dynamically adding, removing and rearranging control functions as well as the addition of entirely new functionality without the need for a fundamental redesign. The component providing this Plug and Play functionality is also referred to as the ACS Framework (ASCF). The ability to take part in these Plug and Play operations is the only true minimum requirement for a node to join an Ambient Network.

This approach provides an excellent starting point when considering the second requirement, namely the feasibility and applicability for the distributed development with a research project. As described above, the integrated prototype developed within AN project consists of implementations of several control functions which are contributed by different working groups of the project. These individual contributions then need to be integrated into one ACS prototype, which calls for an architecture that natively supports the integration and interworking of potentially very heterogeneous components. The modular and dynamic nature of the Plug and Play ACS perfectly suits this need.

B. *Design of the ACS Framework*

The backbone of the prototype design is formed by the decisions on the structure of the ACSF. Note that FE is essentially a logical construct and in general, a node or network element may host one or more FEs. On each node used in the prototype, the ACSF component is available that the different FEs can register to. When an FE needs to use a service offered by another FE in the Ambient Network, it queries the ACSF and is provided with the address of the FE implementing this service. Thus, the individual FEs do not need to know where exactly another FE resides, they really only need to know the well-known address of the ACSF residing on the same node.

For the design of ACSF, it was decided to split the module into two parts. Firstly, a front-end implements an interface that is exposed towards the FEs. This part of the ACSF was specified very early in the development process and has remained relatively stable over time.

The second part, the back-end, is responsible for the ACSF functions as such. This comprises the management of distributed registration information as well as the functionality needed to bootstrap an AN. Through the clear split between front-end and back-end, the implementation of the ACSF can also evolve over time without impacting the interface towards the FEs. E.g., the simple replicated database which was chosen for the first implementation step [4] can easily be integrated with distributed, peer-to-peer based data storage mechanisms which are currently developed within the project.

C. *Intra- and Inter ACS Communications*

As motivated above, the ACS must allow the dynamic deployment of FEs and provide a registration and lookup service for them. Also, FEs will need to dynamically lookup and communicate with other FEs inside the ACS (intra-ACS communication) or even between external ACSs (inter-ACS communication). In order to simplify our architecture, the same communications technology intra- and inter-ACS is used. Due to the dynamic and heterogeneous nature of the AN networking environment, the most important criteria were: fast learning curve, multi-platform support (both OS and programming languages), asynchronous messaging, protocols, security, maturity, performance, internetworking and support for mobile devices. An extensive survey was made and well-known technologies like CORBA, Web Services, OSGi Service Platform, UPnP [4], as well as solutions developed within the project [6] were evaluated. In the end, the list was narrowed down to the most mature technologies from the initial list, CORBA and Web Services, considering that these can also assure a faster learning curve and better multi-platform support.

The two major drawbacks of CORBA in relation to Web Services were the internetworking – CORBA is not able to pass through firewalls and NATs – and the lack of support for mobile devices. On the other hand, Web Services performance is worse than CORBA. Since we are talking about prototyping work and also due to the distributed nature of the ACS, it is mandatory that one can easily deploy a distributed ACS spanning across multiple networks. Another key decision factor was the support for devices like PDAs and smartphones.

These considerations recommended the Web Services as the ideal candidate. Regarding the performance, on the one hand public domain implementations – the ones accessible for prototyping work- are heavy weight and relatively slower. On the other hand there are examples of proprietary solutions running in real time within global telecom operators networks. The prototype must validate the AN solutions in the first place, not to serve as performance benchmarks on global scale. In case that a company opts for introducing these solutions in commercial circuit, then it will certainly allocate the proper resources to procure and optimize a proprietary Web Service product. Weighing all the above arguments, Web Services was chosen for both intra- and inter-ACS communication [7].

IV. REALIZATION OF THE ACSF

A. *The main components of the prototypes*

The modular and dynamic nature of the Plug and Play ACS mentioned in the previous section demands a minimum set of components that enable this approach. The modules that realize this feature are the ACS Registry, the ACS Framework Management Component and the Bootstrapping Manager. These components are present on every node in an AN.

The front-end registry interface exposes a Web Service interface that allows to register/deregister and lookup FEs. The back-end solution consists of a distributed MySQL database.

FEs can be added to the ACS either during the bootstrapping of an Ambient Network or later, when the ACS is already fully functional (complete).

The Bootstrapping Manager monitors which FEs have registered and decides when an AN node or the ACS, respectively, qualifies as complete, i.e. when a minimum set of control functions are available. These functions may be spread over several nodes – in such a case the distributed nature of the ACS becomes apparent. The bootstrapping process is supported by the ACS Framework Management Component that automatically deploys, starts and registers FEs.

Once a FE is registered, it can be looked up by other FEs. Note that one FE may offer multiple services which are identified by Service Access Points (SAP), so lookups may target SAPs, as well. Such a lookup is always done locally, i.e. by querying the ACSF on the same node. However, the database is distributed over all nodes belonging to an AN and it is always kept synchronized. If the requested service (SAP or FE) is not present in the same AN, a Destination Endpoint Exploration Protocol (DEEP) query is invoke, as explained in [6]. It looks up the databases of adjacent ANs, discussed in the following sub-section.

B. *Naming, DEEP and ACSF*

A hierarchical naming scheme is used in the prototype for identifying ANs and their internal elements including entities in the control plane. The level of hierarchy is as follows: Ambient Network: Ambient Network Node: Functional Entity: Service Access Point. Identifiers are created by concatenating two or more names at different levels in descending order, e.g. AN1.ANN2 or FE3.SAP4 etc. There are two key underlying principles. 1) Each level has its own name space. This provides for a flexible naming system as different namespaces can be used at different levels. 2) Names and locators are delinked. The latter is important for two reasons. Firstly, network and control space entities maybe mobile which requires locator changes. Secondly, the locators for different entities are different. For some, such as ANs and AN Nodes, IP addresses are used as locators whereas FEs and SAPs mainly use Webservice URLs. The proposed naming scheme takes care of this heterogeneity and also makes sure that locator changes do not affect identifiers.

To implement such a naming scheme, a flexible name resolution system must be in place. In the prototype, a two-step procedure is applied. When a network/control entity wants to resolve a name, it queries the ACSF which does a lookup in the AN Registry. If the name refers to an entity within the same network, a corresponding record will exist in the local database and hence, the ACSF will serve the query accordingly. Otherwise, if the name refers to a foreign entity, the ACSF uses DEEP to query the foreign AN. The name is sent to the local DEEP server which will send an Explore message to the remote DEEP server which in turn queries its ACSF to find the locator for the received name. Upon finding a match, a Response message is sent to the sender of Explore message. When the Response is received, the ACSF is informed, which then responds to the original query which triggered the resolution process.

C. *HIP and Security*

The mobility and security features of the ACS are implemented using the Host Identity Protocol (HIP) integrated in kernel of the operating system (FreeBSD).

HIP separates the location and the identifier of a host. The binding between the identity of a host and its IP address is dynamic, i.e. the identity remains the same when a host changes its location [8]. Every host using HIP is assigned a Host Identity (HI), a cryptographic identifier that serves as a public key. For the use in HIP packets, Host Identity Tags (HIT)s are created by taking a cryptographic hash over the corresponding Host Identifier. The advantages of using HITs instead of the HIs is that the fixed length makes it easier for protocol coding and the consistent format makes it independent of the cryptographic algorithms used [9].

When two hosts establish a connection for the first time, the HIP base exchange is performed, defined in the HIP base specification [9]. The base exchange serves to make the protocol resistant against Denial-of-Service (DoS) attacks, since it puts a cost of the connection establishment on the requesting entity. Using authenticated packets prevents Man-in-the-Middle attacks [9].

The Security Domain framework [6] implemented in the AN prototype groups and manages resources and also provides secure identification of ANs. A Security Domain hosts a group of resources in one or several nodes according to certain security policies. These policies are defined such that they allow for secure interaction within the group of nodes, or with other nodes in a common way [10].

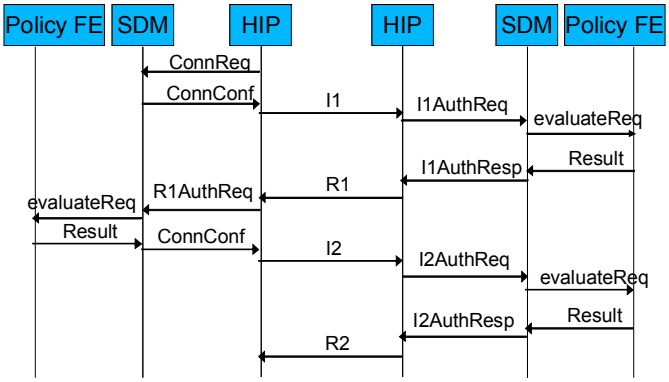

Figure 2: The ACS Security mechanism with HIP

Every Security Domain is managed by a Security Domain Manager (SDM). This entity takes care of assigning certificates to nodes belonging to the same Security Domain (see Figure 2). This certificate is the private key of the public/private key pair that secures the membership of a Security Domain [6]. The policies that regulate the access between nodes belonging to the same Security Domain or between nodes of different Security Domains are managed by a dedicated FE (Policy FE). It uses an Extensible Access Control Markup Language (XACML) implementation. The decisions and information lookups within the policy framework are modeled according to the SICS XACML

reference implementation [11].

D. Triggering and GUI

The triggering Functional Entity (TRG-FE) was initially designed as the AN solution for the distribution of mobility related events [12] [13]. The typical users of the TRG-FE can be considered to be mobility protocols such as Mobile IP (MIP) and HIP. The TRG-FE uses the Web Services model for exchange of mobility events (referred to as triggers). By applying the protocols and description languages of Web Services (SOAP, WSDL), a producer-consumer model is used for distribution of triggers. This means that a producer (HIP, as an example) sends triggers towards TRG-FE, which are distributed further to subscribed consumers of mobility events.

Even though TRG-FE has originally been targeted at the distribution of mobility related events, it turned out to fit the needs of other FE designers, as well. Therefore, it is used also as a distribution engine for general purpose events in the ACSF, i.e., as a notification system.

A visualization framework was also developed to display the network topology, the FEs active within the ACS and the events occurred during the lifetime of the AN. This framework is referred to as ACS GUI, and it is considered as the prototype of a monitoring application. The triggering engine has a distinguished role in feeding the ACS GUI with the events occurring in the ACS.

The relation of the TRG-FE to the ACS GUI and other FEs is described in Figure 3. The GUI from node 3 registers as a consumer to the TRG-FE. Several FEs (grey circles) that populate the ACSs on standalone nodes (node 1 and node 2) are displayed on the left side of the figure. As the FEs produce a trigger, the TRG-FE notifies the GUI about them via XML messages. Then the GUI visualizes these events in a unified framework.

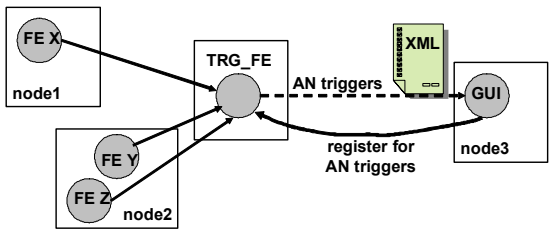

Figure 3: Triggering FE

V. Conclusions

We have described the Ambient Networks approach for developing a common prototype that integrates a vast number of modules and mechanisms in a dynamic and heterogeneous networking environment.

In such large projects it is crucial to focus on the key innovations of the project. For this goal we used a refinement process which started with the definition of Joint Use Cases and ended in the selection of the minimum set of Computational Processes. This approach also yielded an implementation itinerary to follow, where the 'core' modules (Functional Entities, registration and lookup frameworks) were identified that were required by the majority of the teams working on specific concepts. Then we defined the interfaces between this core and the other modules. By this step we halved the time required to develop the prototype, since the core modules were implemented in parallel with the ones that rely on them.

We presented the design process of the AN Control Space Framework and the way we solve the intra- and inter-ACS communication. From the realization process of the ACSF we highlighted the one of the lookup, security and triggering systems. The AN integrated prototype presented in this paper will allow us to test the interoperability of several device types within the ANS. We will also conduct detailed performance testing to assess the scalability of the system. All these steps will further strengthen our work on standardization of the AN concepts.

References

[1] Ambient Networks project homepage, http://www.ambient-networks.org

[2] M. Johnsson, A. Schieder, R. Hancock Eds. "D-A.2: Draft System Description", FP6-CALL4-027662-AN P2/D07-A2, December 2006, URL: http://www.ambient-networks.org

[3] European Commission Sixth Framework Programmes homepage, http://ec.europa.eu/research/fp6/

[4] P. Pääkkönen ed., "D.H.1: Final Application Scenarios and Prototype Design", FP6-CALL4-027662-AN P2/D05-H2, December 2006, URL: http://www.ambient-networks.org

[5] World Wireless Initiative – Ambient Networks demonstrations, IST Mobile Summit Dresden, Germany, June, 2005, URL: http://www.mobilesummit2005.org

[6] "D1.5.: AN Framework Architecture", IST-2002-507134-AN/WP1-D06, December 2005, URL: http://www.ambient-networks.org

[7] Web Services Glossary, W3C Working Group Note 11 February 2004, http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/

[8] HIP for inter.net Project, http://hip4inter.net/

[9] R. Moskowitz, P. Nikander, "Host Identity Protocol (HIP) Architecture", RFC 4423

[10] J. Tenhunen ed. "D1-6: AN Case Studies Report", IST-2002-507134-AN/WP1-D06, 2005, URL: http://www.ambient-networks.org

[11] SICS implementation of the XACML 3.0 draft http://www.sics.se/spot/xacml_3_0.html

[12] Tang H. Eisl J. Eds. "Mobility Support: Design and Specification", FP6-CALL4-027662-AN P2/D9, 2007, URL: http://www.ambient-networks.org

[13] Jukka Mäkelä and Kostas Pentikousis, "Trigger Management Mechanisms for Ambient Networks", International Symposium on Wireless Pervasive Computing, San Juan, Puerto Rico, February 2007.