ns2-MIRACLE: a Modular Framework for Multi-Technology and Cross-Layer Support in Network Simulator 2

Nicola Baldo[*], Federico Maguolo[*], Marco Miozzo[†], Michele Rossi[*†] and Michele Zorzi[*†]
[*]Department of Information Engineering – University of Padova, Italy
[†]Consorzio Ferrara Ricerche – Ferrara, Italy
{baldo,maguolof,rossi,zorzi}@dei.unipd.it, mzzmrc@unife.it

ABSTRACT

In this paper we present a novel framework for ns2 to facilitate the simulation and, in general, the design of beyond 3G networks. The set of libraries we wrote for this purpose is called *Multi InteRfAce Cross Layer Extension* for ns2 (MIRACLE). They enhance the functionalities offered by the Network Simulator ns2 by providing an efficient and embedded engine for handling cross-layer messages and, at the same time, enabling the coexistence of multiple modules within each layer of the protocol stack. For instance, multiple network, link, MAC or physical layers can be specified and used within the same node. The implications of this are manifold. First of all, the framework facilitates the implementation and the simulation of modern communication systems in ns2. Secondly, due to its modularity, the code will be portable, re-usable and extensible.

As an example of the advantages offered by our architecture, we show how the MIRACLE framework can be used to quickly set up protocol architectures for Ambient Networks [1] and evaluate their performance in wireless and multi-technology environments. We stress that, even though the emphasis in the present paper is put on wireless systems, MIRACLE is a general framework which can be used for simulating wired networks as well as a mixture of wired and wireless scenarios. Throughout the paper we also discuss some of the downsides of existing ns2 extensions, which are often programmed in a rather ad hoc manner, according to specific needs or technologies and, as such, are often difficult to extend/re-use. In contrast, our effort aims at providing well defined interfaces and is based on a truly modular architectural design. Our work can be seen as a step toward the definition of a standard framework for the simulation of cross-layer, multi-technology and mobile systems in ns2.

Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]: General, Model Validation and Analysis, Model Development; C.2.6 [**Computer-Communications Networks**]: Internetworking

General Terms

Multi-technology, Cross-layer, Dynamic Libraries, Heterogeneous Networks, Ambient Networks

1. INTRODUCTION

In the last few years, advances in the hardware for wireless networking and, especially, embedded microprocessor technologies have made it possible to manufacture very small radio equipments at low cost. This enables the integration of different technologies in the same mobile equipment. These multi-technology solutions are now available on the market and open up the possibility of exploiting new communication paradigms. As multi-interface hardware becomes available at low cost, there is a parallel need for understanding its performance limits and devising new networking protocols that will make full use of the offered potential. Often, these systems are way too complex to be fully characterized analytically, and we have to resort to accurate simulation tools for their complete understanding. One of the most used simulation tools in the networking research community is without doubt the Network Simulator, *ns2* [2]. We observe, however, that ns2 does not currently support multiple radio interfaces and lacks flexible tools for the cross-layer control of communication systems. Moreover, in the standard distribution of the simulator, the wireless channel is represented via unrealistic models, which may lead to biased results. Also, alternative implementations of the wireless channel are available for specific radio technologies, such as Bluetooth. Nevertheless, these are neither standardized nor re-usable for different radio interfaces. This makes it very difficult to carry out studies on wireless coexistence and spectrum sharing.

In this paper we present an architecture that we developed for the ns2 simulator in order to fill these gaps. Our framework is called *Multi InteRfAce Cross Layer Extension* (MIRACLE) for ns2 [3]. It is conceived as a set of dynamic libraries which are loaded to add support for multi-technology and cross-layering. We also wrote a patch which facilitates the use of dynamic libraries in ns2. Notably, working with dynamic libraries allows the development and subsequent use of new features without the need for re-compiling the whole simulator. In fact, libraries can be loaded on demand at simulation time. Moreover, as we show later, our architecture is highly modular as it allows the interconnection of multiple down and upstream modules at every layer in the protocol stack. Dedicated and broadcast channels are allocated, at each node, for the inter-layer communication of control as well as data messages. We finally observe that, even though our emphasis as well as the examples that we show later in this paper are on wireless systems, the framework can be used to simulate wired networks as well as a mixture of wired and wireless architectures.

In the first part of this paper we introduce MIRACLE, by highlighting its functional structure, the currently available features and how it can be extended for the support of new radio technologies. Subsequently, we give a concrete architectural example on how MIRACLE can be used to model a multi-technology wireless scenario. Simulation results are also reported for selected network settings. The remainder of this paper is structured as follows. In Section 2 we discuss the patch we developed to load dynamic libraries in ns2. In Section 3 we present in detail the MIRACLE framework. In Section 4 and Section 5, we discuss modules ported from ns2 and developed from scratch, respectively.

In order to give an example application of the MIRACLE framework and, at the same time, to demonstrate that it facilitates the implementation and the simulation of beyond 3G wireless systems, in Section 6 we briefly introduce the Ambient Networks (AN) EU funded project. The aim of the AN project is to enable seamless interworking between networks and wireless terminals in a mobile and multi-technology environment. In Section 7, we describe how we implemented the AN architecture in ns2 by means of the MIRACLE framework. In Section 8 we demonstrate, via simulation results, the effectiveness of the MIRACLE approach and discuss the advantages of using it for simulating beyond 3G networks. Finally, in Section 9 we draw our conclusions.

2. PATCH FOR LOADING DYNAMIC LIBRARIES IN NS2

Many researchers around the world are developing modified versions of ns2 in order to introduce new features such as agents, protocols, algorithms, etc. The standard practice adopted in doing this is to get an official version of the ns2 source distribution, make the needed modifications on the source code, add new files somewhere in the existing code tree, and finally build everything into the ns2 executable. In other words, adding functionalities to ns2 means making changes to the whole ns2 distribution. In some cases these changes make their way into the official ns2 project; often, however, this will not happen because of several issues like poor backward compatibility, unproven reliability, and so on. Still, it is often the case that people are interested in using or modifying some of these ns2 extensions (note that there are some very popular ones, such as IEEE802.11e). Installing them involves, in the best case, downloading the official ns2 distribution and patching it [4]. In the worst case, it is necessary to manually replace specific code files in the ns2 code tree [5,6], or even to download an entire modified ns2 distribution [7]. In general, keeping different extensions available requires having separate ns2 installations.

We believe that the introduction of dynamically loadable libraries substantially improves the current way of developing extensions to ns2 and its usability. A list of offered advantages is reported below:

- People can develop add-ons for ns2 (e.g., introducing new agents, packet types, protocols) without having to modify the core simulator.
- New packet headers and types, as well as packet tracers, could be defined to assist debugging, collection of statistics and inter-module communication. These can also be loaded on demand according to user's needs.
- Dynamic libraries can be loaded at simulation time, with no need to recompile the whole ns2 distribution or to keep different ns2 binaries.
- The installation of third-party ns2 extensions is made easier, thereby facilitating their dissemination.
- Dynamic libraries will make life easier for lab technicians and students. In fact, an official ns2 version can be installed by the administrator and students can just build and use their preferred extensions independently.
- Besides, these modifications will make ns2 more modular and scalable. Adding new features to the simulator will be easier and backward compatibility will be preserved.

We observe that dynamic libraries are natively supported in ns2 (see Tcl `load` functionality). However, the set of functionalities which can be accounted for by means of this approach is severely limited by the intrinsic structure of the simulator. As an example, new packet types and headers cannot be added to the code. In order to remove these limitations, we developed a patch enabling dynamic definition of packet types, headers and their corresponding tracers. As mentioned above, this facilitates inter-module communication and collection of statistics. Hence, our patch makes it possible to effectively exploit the benefits of using dynamic libraries thus achieving what we discussed in the bullets above.

Finally, we would like to observe that special care has been taken to ensure backward compatibility: the patch has been designed in order not to interfere with the existing functionalities in ns2. This patch is available at [8].

3. THE NS2-MIRACLE LIBRARY

3.1 Related Work

The main motivation that led us to the development of this library was the need for a flexible and easy to use tool for the simulation of multi-layer and multi-stack architectures in mixed wired/wireless settings. In this respect, there have recently been a few attempts to improve ns2 flexibility, in particular to overcome the current ns2 limit of no more than one wireless interface per mobile node. For example, TENS [9], Hyacinth [10] and the solution proposed by Aguero et al. [11] are extensions to ns2 which introduce the possibility of using multiple wireless interfaces within the same mobile node; however, they are currently limited to the use of a single radio technology (i.e., 802.11) for all interfaces. MW-Node [12], in addition to the support for multiple wireless interfaces, also allows coexistence of different radio technologies and routing protocols within the same node; still, its scope remains somehow limited, since modularity is addressed only at the network layer and below, and the fixed protocol stack architecture imposed by the use of the `MobileNode` is maintained.

We note that the recently started ns3 project [13] shares with MIRACLE some relevant goals, such as enhanced modularity of components; however, there are some major differences between ns3 and MIRACLE. First of all, it is to be acknowledged that ns3 tries to address some ns2 issues, such as support for distributed simulations and emulation, which are not considered in MIRACLE. Nevertheless, this choice for ns3 has required a complete rewrite of the simulator, which prohibits reusability of the many valuable components already implemented for ns2; MIRACLE, on the other hand, can take advantage of reusing ns2 code and can consequently offer many of the features already included in ns2 with little or no development effort (see Section 4). Furthermore, there are significant differences in the node architecture between ns3 and MIRACLE. To the best of our understanding, ns3 adheres to the ns2 concept of having different types of nodes (e.g., internet node, mobile node) with well-defined protocol stack architectures; in this approach, only the developer has the possibility of defining new architectures by writing a new node class. MIRACLE, on the other

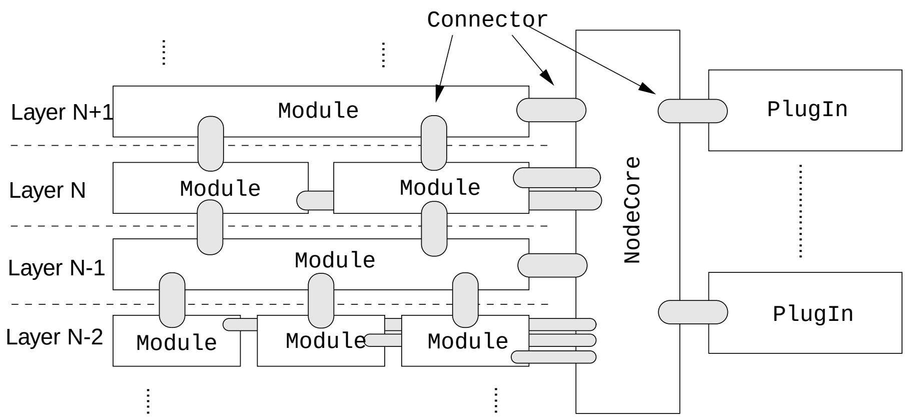


Figure 1: Example of a general multi-layer architecture within the MIRACLE framework.

hand, introduces the concept of a unique, general purpose node which primarily acts as a container for arbitrary protocol stack architectures; this provides the user with great flexibility upon instantiation of the nodes, at the cost of a slight increase in the complexity of simulation scripts. Finally, at the time of this writing it is not very clear to what extent ns3 will provide support for cross-layer interactions, for which MIRACLE provides a dedicated facility.

3.2 MIRACLE Architecture

One of the primary goals of MIRACLE is to facilitate the interconnection of different modules of the protocol stack while, at the same time, uniforming the procedure by which multiple protocol layers are plugged into the same node. We may have, for instance, a node with multiple PHY, MAC or routing layers and we may use all of them in the same simulation by making decisions on which modules to use at runtime.

We started our work from a few existing ns2 classes that were extended to obtain the basic building blocks of our framework. The reason for this choice was to maintain some backward compatibility with previously developed ns2 code (see Section 4). One of the most important blocks is probably the `Module` class. As shown in the left side of Fig. 1, multiple `Module`s can coexist within the same protocol layer and can be connected to up and downstream `Module`s. Dedicated objects, referred to here as `Connector`s, are used for this purpose. Each `Module` contains a specific protocol or entity which may be a PHY, MAC, routing layer, transport protocol, application, etc.

All `Module`s within the same stack are connected to a unique structure called `NodeCore`. The role of the `NodeCore` is twofold. First, it was designed to enable communication among `Module`s and thus to facilitate cross-layer design. The second `NodeCore` functionality consists of managing information and providing functionalities of common interest for all `Module`s.

Regarding cross-layer interactions, we note that the common practice in standard ns2 consists of either including control messages within packet headers or manipulating the ns2 node structure in a rather ad hoc manner. In the former case, however, control messages would be tightly bound to the packet flow whereas, in the latter, it would likely be difficult to re-use/adapt the code to additional needs. Note that these are static solutions as communication interfaces among modules and cross-layer algorithms must be defined in advance, i.e., during the setup of the simulation. In contrast, our solution adheres to the widely accepted concept of having a bus for inter-layer communication [14]. According to our framework, messages among `Module`s can be exchanged at any time and without the need for interleaving them with the data flow. In addition, we standardized how information is exchanged among layers thus achieving modularity and extensibility. In this case, modules can discover each other and communicate with any other entity in the protocol stack at runtime. For instance, the routing layer can discover, during the simulation, which radio interfaces are owned by the terminal. To accomplish this, it is sufficient to send a broadcast control message requiring a response from each available `Module`.

As to the maintenance of common information and functionalities, the `NodeCore` currently maintains the geographical position for each node. To this end, we defined a generic interface which can be used for the implementation of mobility models directly in C++. Currently, the framework features deterministic and Gauss-Markov [15] mobility models.

Another important piece of the architecture is the `PlugIn` class. `PlugIn`s are attached to the `NodeCore` and are the perfect place for cross-layer algorithms: thanks to the `NodeCore`, control messages can be easily exchanged between `PlugIn`s and protocol `Module`s.

Finally, MIRACLE implements a brand new tracing technique: all packets and cross layer messages are traced by each `Connector` as they pass through it. Hence, the development of tracing functionalities is not bound to the implementation of `Module`s. The level of verbosity of message traces is fully tunable and programmable. With tunable we mean that the tracing functionality can be independently turned on/off for each `Connector`. Programmable means that the output of the tracers can be fully defined by the user. Accordingly, each implementation of a `Module/PlugIn` can define its own tracing rules, which can be exploited for debugging or collection of statistics.

A diagram of the MIRACLE architecture is given in Fig. 1.

4. PORTING NS2 MODULES IN THE MIRACLE FRAMEWORK

Special care was taken in the design of our architecture, so as to facilitate the porting of existing ns2 code. In particular, we defined the `Module` class as a child of the `NsObject` class. Hence, we can encapsulate ns2 modules within the MIRACLE `Module` class. This requires redirecting the input and the output of the original ns2 modules to the `Module` class, which is now in charge of connecting the original module with the rest of the protocol stack. This allows the re-use of existing ns2 code. However, in this case modifications to the original ns2 modules amount to re-writing part

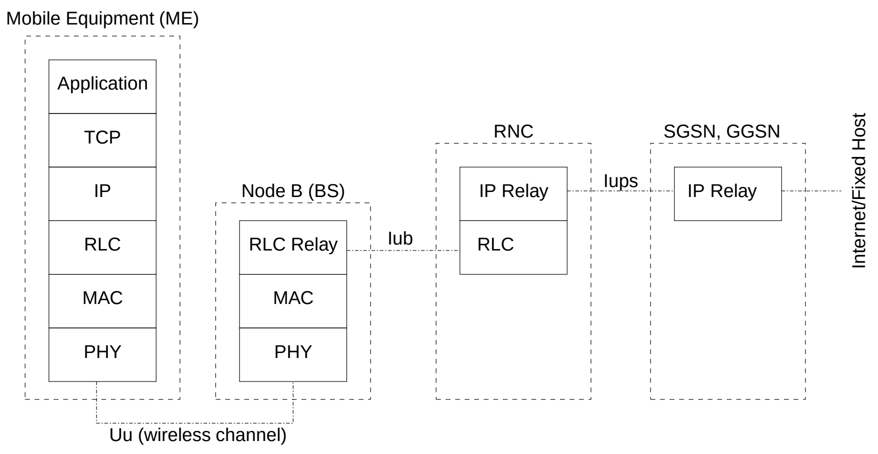


Figure 2: A diagram of the UMTS architecture.

of the original module and re-compiling the whole ns2 distribution in order to make the changes effective. To solve this problem, we can alternatively copy the code of the original ns2 module in the extension of the related MIRACLE `Module` class. In this case, the code is part of the MIRACLE framework and, as explained in Section 2, can be modified and recompiled separately from the rest of the simulator.

As an example, the first procedure was used to port TCP Agents and links, whereas the second one was adopted to port (and extend) the standard IEEE802.11 ns2 module.

4.1 Porting and Refinement of the IEEE802.11 Library

In addition to porting the ns2 IEEE802.11 module to the MIRACLE framework, we added new functionalities such as multi-rate support and a better interference model. In more detail:

- We added support for multiple transmission rates, modulation and coding schemes as defined in the IEEE802.11b/g standards. This includes the possibility of switching the modulation in use at runtime.

- We implemented a realistic interference model which calculates the signal to interference plus noise ratio (SINR) for each connection by considering all packets in flight. The packet error rate is determined as a function of SINR according to packet error rate curves. These for IEEE802.11g are obtained off-line by means of an orthogonal frequency division multiplexing (OFDM) physical layer simulator. For IEEE802.11b, we instead used an analytical model of the direct sequence spread spectrum (DSSS) technique.

- The use of an SINR-based packet error model provides a capture model which is more realistic with respect to the one adopted in ns2, which relies on a pre-determined capture power threshold.

In addition to the above improvements, we also solved some of the bugs reported in [16]: “Direct Access Denial”, “Random Backoff Time” and “Capture Model”[1]. This library is available both as a MIRACLE Module and as ns2 extension [17].

[1]We note that “Capture Model” in [16] actually refers to a synchronization issue, which therefore differs from the capture model issue discussed before in this section.

5. NEW MODULES IN MIRACLE

In this section we present the new modules that we explicitly developed for the MIRACLE framework. We start, in Section 5.1, with the description of a novel library developed to model the physical layer of different radio technologies in a unified manner. Subsequently, in Section 5.2, we describe our UMTS library for MIRACLE.

5.1 MIRACLE Physical Layer Module (MPhy)

One of the most delicate issues in ns2 is the physical layer. Especially for wireless systems, packet errors are often evaluated using simplistic interference models, an example being the standard IEEE802.11 implementation.[2] This led us to the development of the MIRACLE Physical Layer Module (MPhy), which can be used as a basis for the implementation of different radio technologies. MPhy records all packet receptions and gives some basic instruments to calculate the corresponding SINR values. In practice, for all received packets it estimates the SINR as a function of propagation gains, interference due to simultaneous transmissions and correlation factor due to channel overlapping.[3] All MPhy components are implemented as parent-classes which can be extended to the specific technology in use. In detail, in addition to the standard ns2 free space and two ray ground reflection models, we developed a full propagation model accounting for path loss, shadowing and multi-path fading phenomena: the path loss is calculated according to the well known Hata model [18], shadowing is tracked by means of the Gudmonson model [19] and fading is simulated for each link through a Jakes simulator [20] with a programmable number of oscillators. The improvements in [21] were also considered in order to enhance the goodness of the Jakes simulator in the presence of multiple users. Finally, SINR values are translated into packet errors by accounting for the specific modulation and error correcting code adopted by the technology under consideration. This last process, which is technology dependent, is done by extending the MPhy class and implementing the needed error model.

5.2 UMTS Library

The UMTS library was developed starting from MPhy (see Section 5.1) and the *eurane* extension for ns2 [4]. Thanks to MPhy

[2]Whenever two nodes transmit in parallel, a *rate-independent* power threshold is used to assess the correctness of the reception.
[3]With channel overlapping we mean channels using non fully orthogonal frequency bands.

we implemented a physical layer accounting for multi-user interference, power control and spreading/scrambling operations. SINR measurements are translated into packet errors using suitable approximations, which we calculated off-line (similar to the fittings in [22]). This increases simulation speed while preserving the required accuracy. Regarding radio link control (RLC) features, we ported the acknowledge mode (AM) RLC from eurane, which implements packet fragmentation, selective repeat ARQ (with a bitmap acknowledge mode) and data concatenation. We added the SDU discard functionality in order to avoid infinite retransmission loops (as is often done in practical systems). An example of the UMTS architecture is given in Fig. 2, where each block is obtained as an extension of the `Module` class (see Fig. 1). For readability, in this figure we do not explicitly mention the `NodeCore` class, which is always required to correctly configure a node.

6. AMBIENT NETWORKS FRAMEWORK AND MIRACLE

MIRACLE was used to implement, in ns2, the framework developed within the EU funded Ambient Networks (AN) project [1, 23, 24]. The AN project targets transparent wireless access and services in a multi-technology environment. One of the main objectives of the project is to provide support for multi-technology terminals, i.e., to allow users to seamlessly migrate between different technologies and networks and, in addition, dynamically manage the business relations with their access providers. The key goal is to provide users with the services they want irrespective of their location. This is achieved through cooperation between networks. We note that, in a mobile environment like the one envisioned in the Ambient Networks project, cooperation has to be established "on the fly". The current version of ns2 is not adequate for the simulation of these types of systems. First of all, simultaneous usage of multiple wireless technologies is not natively supported by ns2 `MobileNode`s. In addition, a coherent architecture for exchanging control messages, switching between access interfaces and, in general, enabling cooperation at every layer of the protocol stack is still lacking. Finally, cross-layer solutions are often implemented, as discussed earlier in this paper, through programming tricks such as piggybacking control messages within data packets. This, however, might lead to wrong results and is neither portable nor re-usable as we change the technology in use. Our MIRACLE framework was conceived to fill these gaps and thus to facilitate the design, the development and the simulation of next generation wireless systems. In what follows, we go through the main concepts of the AN projects by giving particular emphasis to the system architecture for multi-technology support and its functional elements, and explaining how it can be exploited to realize the AN vision discussed above. In doing this, we constantly refer to the MIRACLE library and on how the architecture in question can be realized through its use. In Section 7 we discuss how the AN architecture was implemented in ns2 using MIRACLE. Finally, in Section 8 we report simulation results.

6.1 Overview of Network Composition

One of the most important concepts developed within the AN project consists of the so called *Network Composition.* Network Composition is a dynamic, automatic and uniform framework that allows cooperation among networks. It can be exploited to enable, e.g., users to access a new network or to stay in the same network but change the access technology in use (as, for instance, they detected an access opportunity at lower cost). In addition, Composition can be used to modify business relations with the access provider, change security parameters/profile and so on. According to the AN framework, two AN-aware networks can communicate only after a successful Composition procedure. Of course, backward compatibility with AN-unaware systems is preserved. However, Composition procedures, if present, allow a full integration between networks and terminals, that are then able to move across systems (and technologies) and to exploit the full set of functionalities offered by the visited Ambient Networks. Once in place, a Composition relation is described by a Composition Agreement (CA) between the networks or parties. In addition, different levels and types of co-operation are supported, e.g., network attachment of user devices, configuration of Personal Area Networks (PANs) and joint resource control of large operator networks. Also, dynamic roaming should be supported, i.e., situations where the user or the home operator do not have any previous agreement or relation with the operator of the visited network and therefore an agreement needs to be established before the user can connect. The Composition process consists of five phases: Media Sense, Discovery and Advertisement, Network Attachment, CA Negotiation and CA Realization. Further details on the AN architecture are discussed in the following Section 6.2.

6.2 A Modular Architecture for Multi Technology Support in Wireless Systems

In Fig. 3, we report a diagram showing the protocol architecture for an Ambient Networks enabled terminal. For illustration, UMTS and IEEE 802.11 radio technologies are plotted in the figure. A standard IP-enabled protocol stack is used as a starting point. In addition, we account for a number of modules (shown in the right-side of the figure) which contain the Ambient Networks intelligence. For instance, these modules are responsible for initiating a CA between the mobile entity and the selected Access Point (AP), to monitor the connectivity status of the terminal, to change the wireless technology in use, etc. Ambient Networks modules are referred to as Functional Entities (FEs). The Generic Link Layer FE (GLL in the AN terminology) is an adaptation layer which is interposed between the IP layer and the technology dependent layers, i.e., the Link Layer (LL) and the PHYsical layer (PHY). Its main role is to enhance existing functionalities at the link layers of the owned radio technologies. For instance, thanks to the GLL one could change/monitor LL parameters, add new features such as Hybrid ARQ algorithms, packet based forward error correction (for enhanced multicast performance), etc. In addition, as shown in Fig. 3 the GLL is in charge of obtaining QoS indicators for both the MAC and the PHY layers. These indicators are either obtained through the reception and the subsequent elaboration of the advertisements sent by the APs or from the collection of statistics, such as bit error rate and received power, during data transmission/reception. Quality indicators may be specifically related to the received power or be user defined (indicated, in Fig. 3, by Pow/SINR and QoS, respectively). These quality indicators are then passed to the Multi Radio Resource Management FE (MRRM). MRRM contains the Network Advertisement FE (NAD) and an execution logic which is MRRM specific. The MRRM might be seen as the heart of the AN architecture: it makes decisions on the APs to join and the radio technologies to use at any time, it sets parameters to achieve energy savings and, in general, to optimize the performance for the currently used network interfaces. In detail, the role of the NAD is to decode incoming advertisements (from neighboring APs) as well as to put the advertisements to be sent by the terminal in the right AN format. All FEs are connected to the Generic Transport Layer Protocol (GTLP), which is the transport protocol used for inter-terminal AN communication.

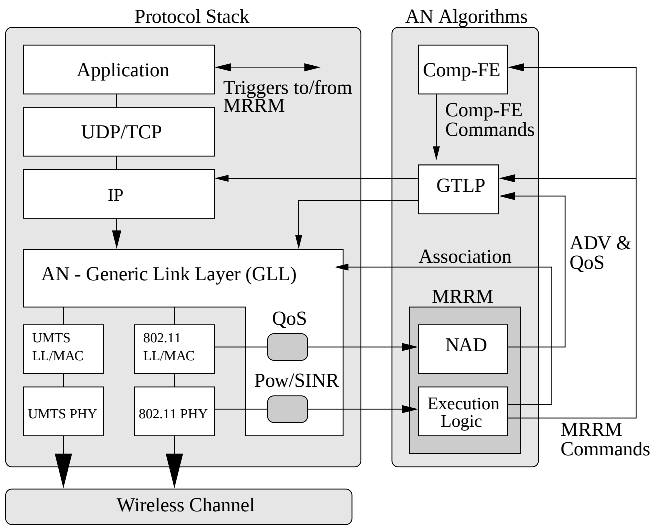


Figure 3: Diagram showing the protocol architecture for an Ambient Networks enabled terminal.

Note that the MRRM (or better its *execution logic*) is also directly connected to the protocol stack. In particular, the execution logic of the MRRM is directly attached to the link layer (via GLL). This is to handle network association procedures (such as Wi-Fi association messages) when IP connectivity still has to be established. In addition, LL messages are required by Ambient Network Attachment Procedures, which are necessary to establish basic AN connectivity with a foreign network. These messages are also managed by the MRRM execution logic. To sum up, there are two levels of communication. The first (and simplest) level exploits LL connectivity, and is used to send advertisements (from APs), network attachment requests (from mobile terminals), to associate a mobile nodes with a new network and, finally, to execute Ambient Networks attachment procedures. The second type of communication, which is AN specific, happens through the GTLP, which is in charge of exchanging AN-messages between different physical entities (e.g., terminal and AP). However, this last type of communication requires IP connectivity and for this reason cannot be managed at the link layer. The last component in the architecture is the Comp-FE, which handles Composition procedures. Comp-FE is connected to the execution logic in the MRRM and communicates to pair Comp-FEs through the GTLP. Composition is not treated in detail in this paper, for further information the reader is referred to [24, 25]. Tables including currently joined APs (or networks), corresponding QoS and the type of Composition which is currently active for each of them are stored at the MRRM and at the Comp-FE. We observe that this structure is rather general, and can be easily extended by adding more FEs. Finally, note that the AN intelligence reacts to triggers from the application layer, which sets QoS reference profiles and indicates whether the user is actually satisfied about the perceived quality.

We stress that in this paper we focus on a specific AN architecture which is, however, far from being complete or representative of the whole Ambient Networks framework. In the AN project, in fact, many more FEs are being defined and additional aspects, such as authentication, billing, security, media server support, etc., are considered as well. These are generally implemented as FEs and have well defined roles and interrelations. In this paper, we instead concentrate on the minimal level of detail we need to evaluate handover procedures in networks featuring multiple technologies.

7. REALIZATION OF THE AN ARCHITECTURE IN MIRACLE

We will now describe how the AN architecture outlined in the previous section has been simulated using the MIRACLE framework. First of all, MIRACLE multi-interface support was exploited to create a mobile terminal equipped with both an 802.11 and a UMTS interface. To simulate these wireless technologies, we used the modules presented in Sections 4.1 and 5.2, respectively. Furthermore, the flexibility of the node architecture in MIRACLE allowed us to implement the GLL as a `Module`, and interpose it between the IP and the Link Layer. We point out that doing this in ns2 would have required either modifications to the 802.11 and UMTS link layers so as to add the necessary functionalities, or a substantial rewriting of the `MobileNode` architecture (to add a new layer between IP and LL). Similarly, MIRACLE `PlugIn`s were exploited for the implementation of the FEs which make up the AN intelligence and do not have a precise placement in the protocol stack but, rather, are meant to interoperate with all protocol layers. Thanks to the Cross-layer communication facility provided by the `NodeCore`, FEs are allowed to communicate among themselves, as well as with all `Module`s in the protocol stack; this is exploited, for instance, to exchange QoS-related information and to perform power measurements. We stress that exploiting MIRACLE functionalities the development of the cross-layer communication system was rather quick, whereas using standard ns2 would have required a great effort in finding tricks and hacks to exchange the needed messages among protocol layers and additional software modules implementing the FEs. Finally, the flexibility of the MIRACLE framework allowed us to go even beyond the cross-layer approach and to implement with ease some cross-layer/cross-device solutions with different levels of hierarchy. For instance, as shown in Fig. 3, the Execution Logic can communicate with its peer at a different node thanks to the the GTLP, which wraps AN messages into IP packets and controls their transmission over the network; this entails some degree of hierarchy between MRRM and GTLP.

8. VALIDATION THROUGH SIMULATION RESULTS

In the next two sections, we describe the simulation scenario as well as the essence of the Execution Logic we considered at the MRRM (Section 8.1) and we finally report some simulation results (Section 8.2).

8.1 Simulation Scenario

To numerically evaluate the correctness and, at the same time, the effectiveness of our approach, we set up a simulation scenario as follows. We considered two radio technologies, i.e., IEEE 802.11g and UMTS. Mobile terminals are randomly scattered, at the beginning of the simulation, within an area of 400×400 m^2. We consider a single UMTS AP, placed in the center of the area so as to give coverage to all nodes. We additionally considered a single IEEE 802.11g AP, also placed in the center of the simulation area and providing coverage for the terminals placed within a distance of about 100 m. All nodes are equipped with both radio technologies and are mobile. For the physical mobility, we adopted the Gauss-Markov mobility model [15] considering the two average speeds of 2 and 15 Km/h. A schematic representation of the simulation scenario is shown in Fig. 4. Terminals receive UDP data traffic from their Mobile Network Operator (MNO in the figure), which is placed in the fixed Internet portion of the network. The MNO is connected to both UMTS and 802.11 APs via two dedicated wired channels. These are error free, fixed delay (200 ms) channels. Data flows in the downlink direction (APs $\rightarrow$ terminals), whereas standard (e.g., AP association, ARP, etc.) and AN signaling messages use both uplink and downlink channels. Downlink data traffic is bursty: during each burst, data is sent continuously at a rate of 70 Kbps. In our simulations, we considered burst durations of $5, 10, 15, 20, 25$ and 30 s. The inter-burst interval is constant and it was set to 10 seconds for the simulation results we report in this paper.

For the radio access management policy, we considered the following two cases. As a basis for our performance evaluation, we implemented an Ambient Networks unaware system, where UMTS and 802.11 coexist but extremely simple rules are used to select the radio access. In particular, in this case the radio technology is selected at the beginning of each burst, based on received signal levels, and is kept unchanged until its end. As a second solution, we considered an Ambient Networks aware selection policy, according to which available access opportunities are continuously evaluated and handover between systems is possible at any time. In this case, the decision making engine is placed in the MRRM and classical methods are used to mitigate unnecessary handovers and ping-ponging.[4] In the simple access selection policy no action is taken when the data channel is idle (i.e., the radio in use is evaluated and possibly changed only during the reception of a burst of data), while, in the AN-enabled policy, MRRM continuously monitors the perceived attachments in order to be ready as soon as the data will come.

Finally, in our simulations relaying was not permitted, i.e., terminals could only communicate directly with any of the APs and could not exploit other terminals as repeaters.

8.2 Simulation Results

Simulation results are shown in Fig. 5. In Fig. 5(a) we report the average packet error rate that a terminal experiences during the reception of a burst of data. Both the simple and the AN-enabled access selection policies (see previous section) are shown in the figure. As expected, AN algorithms provide better performance for all burst lengths. This is due to the careful and continuous assessment of available accesses. In addition, we note that performance increases at lower speeds (2 Km/h in the figure). This is due to the fact that an increased velocity corresponds to a higher number of handovers which, in turn, may lead to a higher packet loss. We note that, however, the increase in the packet error rate is limited.

Fig. 5(b) shows the total number of control bytes transmitted by an Ambient Networks terminal during the reception of a burst of data. First of all, we observe that an increasing speed leads to an increased traffic. The reason is the same we discussed above, i.e., a higher handover frequency. Secondly, the control traffic overhead grows with the burst duration. In fact, users during longer bursts are more likely to move out of coverage of the serving AP. In this case, they must initiate a new handover procedure, thereby generating further control traffic.

9. CONCLUSIONS

In this paper we presented the *Multi InteRfAce Cross Layer Extension* for ns2 (MIRACLE). This framework allows the extension of the protocol stack so as to add multiple protocols within each layer and, in addition, facilitate the design of cross-layer algorithms through the definition of a dedicated communication bus.

[4]In practice, triggers from the PHY are received at the MRRM whenever the signal strength decreases below system dependent thresholds and hysteresis loops are accounted for to avoid ping-ponging between radio technologies. The same thresholds are used for the evaluation of the two radio access management policies.

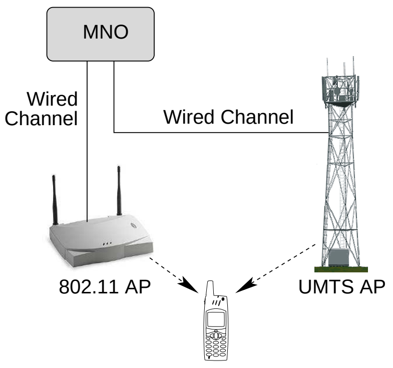


Figure 4: Simulation scenario.

In the first part of the paper we discussed pros and cons of existing ns2 architectures supporting multiple access techniques. We subsequently described our framework, discussing the libraries we developed and their advantages with respect to previous solutions. These include the support for 802.11 and UMTS radio technologies as well as a generic physical layer `Module` that we use to characterize the transmission over the wireless medium. As an example of the advantages offered by our architecture, we then showed how MIRACLE can be used to quickly set up protocol architectures for Ambient Networking and evaluate their performance in wireless and multi-technology environments. We stress that, even though the emphasis in the present paper is put on wireless systems, MIRACLE is a general framework which can be used to simulate wired networks as well as a mixture of wired and wireless scenarios. Our work can be seen as a step toward the definition of a standard framework for the simulation of cross-layer, multi-technology and mobile systems in ns2.

The work on the Miracle library is still ongoing. In more detail, we would like to improve the tracing functionality in order to avoid excessively large trace files and to allow filtering according to specific keywords. In addition, a number of extensions are possible, e.g., to port existing ns2 routing protocols and to implement further radio technologies such as IEEE802.15.4, WiMAX, Bluetooth by, e.g., adapting their standard ns2 implementations.

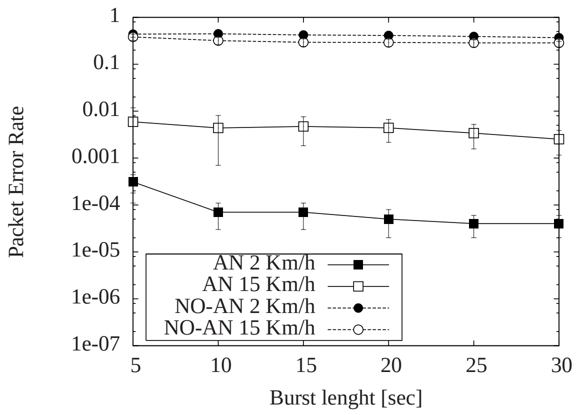
10. ACKNOWLEDGMENTS

This article is a product of Ambient Networks, a research project partly supported by the European Commission under its Sixth Framework Program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Ambient Networks project or the European Commission. We thank all partners involved in WPH task 3 of the Ambient Networks project for the thorough discussions on the Ambient Networks architecture. In addition, we would like to thank Matteo Trivellato for his support with the characterization of the IEEE802.11g OFDM physical layer.

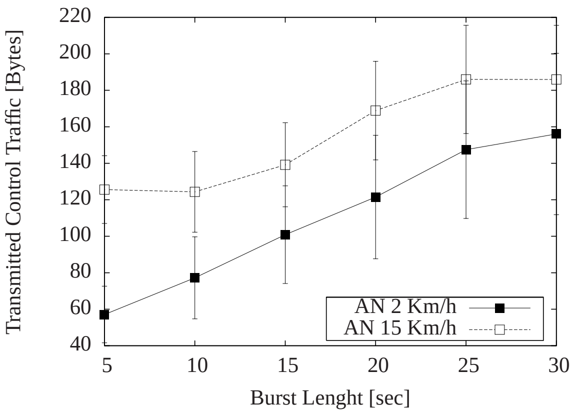
11. REFERENCES

[1] N. Niebert, A. Schieder, H. Abramowicz, G. Malmgren, J. Sachs, U. Horn, C. Prehofer, and H. Karl, “Ambient Networks: an Architecture for Communication Networks Beyond 3G,” *IEEE Wireless Commun.*, vol. 11, no. 2, pp. 14–22, Apr. 2004.

[2] The Network Simulator - ns2, http://www.isi.edu/nsnam/ns/.



(a) Average packet error rate for simple an AN-aware policies.



(b) Average transmitted control traffic for AN-aware policies.

Figure 5: Performance of simple and AN-aware access selection policies: (a) performance comparison in terms of average packet error rate for simple (NO-AN) and AN-aware (AN) access selection policies; (b) average control traffic transmitted by AN handover algorithms. 95% confidence intervals are plotted by means of vertical bars.

[3] ns-MIRACLE: Multi InteRfAce Cross Layer Extension for ns-2, http://www.dei.unipd.it/ricerca/signet/tools/nsmiracle.

[4] Enhanced UMTS Radio Network Extensions for ns-2, http://www.ti-wmc.nl/eurane/.

[5] An IEEE 802.11e EDCA and CFB Simulation Model for ns-2, http://www.tkn.tu-berlin.de/research/802.11e_ns2/readme_EDCA.txt.

[6] NO Ad-Hoc Routing Agent (NOAH), http://icapeople.epfl.ch/widmer/uwb/ns-2/noah/.

[7] New ns-2 802.11 support, http://yans.inria.fr/ns-2-80211/.

[8] Patch for Loading Dynamic Modules in ns-2, http://www.dei.unipd.it/ricerca/signet/tools/ns_dl_patch.

[9] The Enhanced Network Simulator, http://www.cse.iitk.ac.in/users/braman/tens.

[10] Hyacinth: An IEEE802.11-based Multi-channel Wireless Mesh Network, http://www.ecsl.cs.sunysb.edu/multichannel.

[11] R. Agüero and J. Pérez, "Adding Multiple Interface Support in NS-2," Available at http://personales.unican.es/aguerocr.

[12] L. Paquereau and B. E. Helvik, "A Module-based Wireless Node for ns-2," in *Proceedings of the first Workshop on NS2, Pisa, Italy*, 2006.

[13] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley, "ns-3 Project Goals," in *Proceedings of the first Workshop on NS2, Pisa, Italy*, 2006.

[14] S. Shakkottai, T. Rappaport, and P. Karlsson, "Cross-layer design for wireless networks," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 74–80, Oct. 2003.

[15] B. Liang and Z. Haas, "Predictive distance-based mobility management for PCS networks," in *Proceedings of IEEE Infocom*, New York, NY, US, Mar. 1999.

[16] I. Purushotaman and S. Roy, "IEEE802.11 implementation Issues in Network Simulator 2," Available at http://ee.washington.edu/research/funlab/, Dept. of Electrical Engineering, University of Washington, US.

[17] An improved 802.11 implementation for ns2 with enhanced interference model, http://www.dei.unipd.it/ricerca/signet/tools/dei80211mr.

[18] A. Goldsmith, *Wireless Communications*. Cambridge University Press, 2005.

[19] G. L. Stüber, *Principles of Mobile Communication*. Kluwer Academic Publishers, 2003.

[20] W. C. Jakes, *Microwave Mobile Communications*. Wiley–IEEE Press, 1994.

[21] P. Dent, G. E. Bottomley, and T. Croft, "Jakes Fading Model Revisited," *IEEE Electronics Letters*, vol. 29, no. 13, pp. 1162–1163, June 1993.

[22] M. Rossi, P. Casari, M. Levorato, and M. Zorzi, "Multicast Streaming over 3G Cellular Networks through Multi-Channel Transmissions: Proposals and Performance Evaluation," in *Proceedings of IEEE WCNC*, New Orleans, Lousiana, US, Mar. 2005.

[23] The Ambient Networks Project, http://www.ambient-networks.org/.

[24] N. Niebert, A. Schieder, J. Zander, and R. Hancock, *Ambient Networks*. John Wiley & Sons, 2007.

[25] C. Kappler, P. Pöyhönen, M. Johnsson and S. Schmid, "Dynamic Network Composition for Beyond 3G Networks: a 3GPP Viewpoint," *IEEE Networks*, vol. 21, no. 1, pp. 47–52, Jan/Feb 2007.