Discovering Points of Interests Through A Web 2.0, Location-based Architecture

Stefano Ferretti, Vittorio Ghini
Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7, 40127, Bologna Italy
{sferrett, ghini}@cs.unibo.it

Abstract

We propose a distributed architecture for the support and the deployment of location-based, discovery services in mobile contexts. Web 2.0 technologies are utilized to locate point of interests for the mobile user. Such discovery is based on a mashup of services that queries different yellowpages-like sites, filters and merges the returned items, and exports the final list to be sent to the mobile client. A cross-layer networking technique is employed on the wireless leg. This offers to the user a seamless navigation, while moving through different and heterogeneous wireless networks. Importantly, our approach also balances the network traffic over multiple wireless networks. Results from a real experimental assessment confirm the efficacy of the proposed approach.

Keywords: Cross-layers architectures, Web 2.0, context-aware applications.

1. Introduction

Discovering Internet services through mobile devices while on the move is becoming a frequent occurrence for many users. The idea is to query context-aware services, by exploiting the ability of current devices of tracking the geographical location of the user [2],[3],[4],[5],[6]. The most prominent example of these types of applications consists in locating nearby places of interest, such as, for instance, restaurants, pubs, hotels, banking cash machines, shops.

The fact is that if one wants to make such applications really reliable and able to seamlessy operate, they should be accompanied with networking solutions able to offer always-connected services. In particular, due to the many available networking technologies, the diverse coverage of these networks, the different performances these may offer, the different costs to connect and exploit them, strategies would be needed to actively manage all available networks and dynamically create connections using only those that may optimize the communication (in terms of throughput, access fees, reliability, etc.).

Novel applications have been recently developed which allow to retrieve information on the geographical area where the user is currently located. This increasing market is further boosting the demand for novel location-based technologies able to provide detailed geographical data. Each of these exploits different localization techniques, ranging from the use of classic GPS, methods exploiting information on the reachable cells of cellular networks, Wi-Fi positioning systems, as well as hybrid solutions.

Google Mobile is probably the most prominent example of location-based application, which offers to users the possibility to interact with the Maps service of Google on mobile terminals [7]. Users may be located using EOTD (Enhanced Observed Time Difference) and TDOA (Time Difference Of Arrival) technologies. Other examples are various and simple applications on sale for cell-phones, e.g., AroundMe for iPhone [8]. Many of these applications, especially those specifically built for iPhone, primarily use the Wi-Fi Positioning System developed by Skyhook Wireless [9]. Using the information on MAC addresses of nearby access points, this system is able to locate the user with an error of 20-30 meters. The key of the mechanism is probably the database Skyhook manages, which includes more than 100 million Wi-Fi access points, with their related geographical position [10].

All these approaches offer viable solutions, which are exploited in real applications. There are some limitations, however. First, these applications must be installed on the device (i.e., they are not Web-based).

Second, they usually refer to a single repository of information to locate point of interests (e.g., Google

Maps), without resorting to the wide amount of information available on different Web services. In substance, these applications do not exploit at all Web 2.0 technologies and the mashup of Web services, which are instead widely utilized by applications offered to traditional (static) users, and have become fundamental to discover data and news in the wired Internet [11].

Third, the operating system and its implementation of networking protocols on mobile terminals usually impose the use of a single network for the communication. Hence, the typical nomadism of the user is not supported by an adaptive and seamless exploitation of different networks, in order to guarantee an always best connected service.

Based on these considerations, we have developed a cross-layer system for the dynamic discovery and localization of points of interest near the actual position of a given user [1]. The proposed approach is a mixture of Web technologies and cross-layer networking techniques. The user simply exploits a browser. A Web service mashup has been developed that, by exploiting Web 2.0 based techniques, queries different yellopages-like Web sites, which in turn are able to identify places of a certain type together with their addresses. The mashup is based on Dapper [12], Yahoo Pipes [13], Google Maps [14] and Reverse Geocoding [15].

We assume that the user has a Mobile Device (MD) equipped with multiple heterogeneous wireless network interfaces to connect to different access networks. This assumption is quite reasonable, having in mind the technical characteristics of current cell-phones and PDAs. In particular, in the presented scenario we assume that the MDs incorporate two or more Wi-Fi network adapters (IEEE802.11a/b/g/n) that offer connectivity on a local geographical scale, and one wireless network adapter that offers connectivity on a large geographical scale by using a long-range communication technology (e.g., GPRS/EDGE, UMTS, 3G/HDSPA, Mobile WiMAX).

When multiple network interfaces are available at a given mobile terminal, there are several important considerations worth of mention. On one hand, with respect to Wi-Fi local-range communication technologies, those that offer connectivity on a large geographical scale, such as GPRS/EDGE, UMTS, 3G/HDSPA, Mobile WiMAX and others, share the following disadvantages: i) a lower available bandwidth, ii) a larger energy consumption (especially during transmissions) and, iii) generally, an higher access fee.

On the other hand, the Wi-Fi technology provides connectivity only within a limited coverage range (150-300 m); hence, it is likely that a nomadic user that walks around the city will enter and leave different areas where different IEEE802.11a/b/g/n Access Points (APs) are active. As a consequence, a Wi-Fi network interface card may perform several handoffs. This continuous association and de-association to different APs may impose that the terminal changes its IP address. A main consequence of this undesired situation is that all previous running applications must be re-started or reconfigured.

The approach we have developed allows to overcome these limitations. It is based on the idea of concurrently exploiting different, heterogeneous and possibly independently managed wireless network infrastructures. This is accomplished through the use of a user-driven cross-layer architecture, that provides a simultaneous use of available wireless network interfaces on the mobile device. This allows to meet application QoS (quality of service) requirements (i.e., responsiveness, reliability, availability, continuity of the communications) and guarantee the viability of services offered by our mashup.

In view of the considerations above, the heuristics we use consists in exploiting the long-range communication technology only when strictly necessary, thus preferring short-range communication technologies, so as to i) limit the electromagnetic energy to which the user is exposed, ii) guarantee a longer battery duration, iii) limit the costs of the communications, and iv) exploit the overall bandwidth provided by the available APs, without lack of communications continuity.

It is important to notice that the use of an underlying service able to maintain a connection active, even in presence of horizontal or vertical handoffs, is of paramount importance for the success of the application we present in this work. Indeed, the mashup takes a while (20-30 secs) to query all different Web services, retrieve the information requested by the user from these different services, based on the user's geographical position, merge such data and encode it as a list of items based on the JavaScript Object Notation (JSON). During such time period, the connection between the client and the mashup service must remain open. Results from an experimental assessment we performed confirm the viability and the efficacy of the presented proposal.

This paper extends the work presented in [1] in several ways. A complete background section has been added that discusses host mobility and location-based services issues. A deep description of the design and the implementation of the proposed architecture is provided. Moreover, a set of experimental results that validate our proposal is reported with a related discussion.

The remainder of this paper is organized as follows.

Section 2 reports a state of the art of the existing systems which share the same final objectives of our approach. Section 3 describes the system architecture we devised. Section 4 discusses details related to the Web mashup built to dynamically discover points of interest near a given user. In Section 5 we analyze details on the cross-layer architecture. Section 6 reports on the experimental evaluation of the proposed system we performed. Finally, Section 7 provides some concluding remarks.

2. Background and State of the Art

In this Section, we review the state of the art with respect to location-based services that exploit features of Web 2.0, thus highlighting the possibilities which result from the use of location-based services into novel Web applications. Then, we survey the basic design principles shared among the most important architectures and protocols devoted to the mobility management. When describing all these proposals, a distinction is made to locate them in the ISO/OSI network stack (see Figure 1).

2.1. Web 2.0 Location-based Services

Location-based services represent the new frontier for novel Web 2.0 applications. There are several examples worthy of mention. We already cited the classic Web 2.0 applications that provide maps and related information on the geographical area where the user is placed. Google Maps, MapQuest, Yahoo! Maps are just few examples. The interesting thing is that these services allow 3rd party developers to exploit their features and contents, through the use of open APIs. This facilitates the creation of novel services.

When location-based systems are integrated with social applications, novel services can be provided that allow to track not only the position of the single user, but also that of his/her friends. Loopt, Pelago, Moximity, BrightKite, GyPSii, Citysense, Whrrl, Plazes are just few examples of applications. It is worth mentioning that these applications have not received the same interest that users show for the traditional social applications, e.g. Facebook and Twitter. Maybe, this is due to the fact a general user would prefer not divulgating his/her actual position to all his/her contacts.

Another interesting software technology is represented by GeoRSS and Geotagging techniques. GeoRSS exploit Really Simple Syndications (RSSs) for identifying a geographical data as a RSS feed. More specifically, GeoRSSs extend existing feeds with geographic information. This would allow, for example, to post an information in a blog about some point of interest and map directly the mentioned locations [16].

Geotags are used to add geographical information (e.g., longitude, latitude) to websites, images, RSS feeds, videos and more. The idea is that Web services, applications and users then can query this information to obtain directions, for instance. Geotags differ from a simple address in that they usually are encoded in metadata and are not visible as part of the Web page.

2.2. Mobility Architectures

The aim of a mobility architecture is to ensure a MD may move across access networks seamlessly accessing network services. Ideally, a mobility architecture is responsible for i) unequivocally identifying each given MD, ii) allowing each MD be reachable from its correspondent nodes, iii) monitoring the QoS channels to predict the need of a handoff, and iv) performing the handoff seamlessly, thus ensuring the continuity of the communications.

Almost all of the mobility management approaches follows the Always Best Connected (ABC) model [17] that suggests the selection of a preferred Network Interface Card (NIC) to be used as single point of access to the Internet, until the performance degrades too much. A two-steps configuration procedure is used: firstly, each MD's NIC detects the best network access point, from those available, and connects to it; in the second step, the MD elects its preferred NIC from those available and, from this instant, the MD uses the wireless network connected with the preferred NIC as its single point of access to the Internet. When the performance of the preferred NIC degrades, the MD detects a new preferred NIC (and the wireless network connected with it) that replaces the previous one, by means of a handover. This is the idea of networks integration that is at the basis of the so-called fourth-generation (4G) networks. This approach prevents the simultaneous utilization of all the MD's NICs and is therefore unable to take advantage, in terms of QoS and costs, of their different characteristics.

In IP-based communications, the MD's IP address plays the twofold role of MD's identifier, as it distinguishes uniquely the MD, and MD's locator, as it identifies the position of the MD in the network. When a MD moves in a different network, it cannot maintain its IP address; thus, the MD acquires a new IP address from the new network, loosing its identity; hence, it is forced to inform its correspondent nodes (CNs) of its new identity and location. More generally, a mobility management architecture must allow two hosts communicate between them even when both may move and change simultaneously their addresses. In fact, in

such a situation, each MD would be unable to contact its CN as it does not know the CN's new IP address.

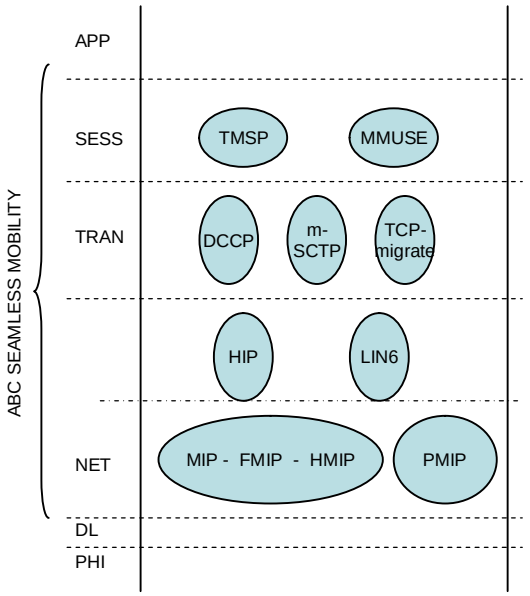

Figure 1: Architectures for mobility management

To solve this problem, all the mobility management architectures adopt, at different architectural layers, a general solution based on two principles: i) defining a unique MD's identifier, independent from the MD location, and ii) providing a localization service, always reachable from the CN, that maintains a mapping between MD identifier and MD current locator, even when the MD moves. Such a location service is usually composed of a sort of Location Registry (LR) working on a server having an IP address which is static, public and known. When the MD changes its IP address, it informs the LR with a registration phase. This way, since a CN knows the MD's unique identifier, when it wants to initiate a new communication with the MD or it wants to continue a communication with the MD that has chanced its address, the CN starts a lookup phase and asks the location registry for the MD's current address. After that, CN may use the obtained MD's address to directly contact it.

Network-layer mobility architectures can be classified in two categories, host-based and network-based, depending on who is the entity that manages the signaling. Host-based architectures, such as Mobile IP version 6 (MIPv6) [18] and its optimizations Fast Mobile IPv6 (FMIP) [19] and Hierarchical MIPv6 (HMIP) [20] add an entity in the architecture, referred as the Home Agent. Such agent acts inside the access network from which the MD belongs, and basically plays the role of LR. A mobility protocol stack, implemented within the MD, is used to generate and manage the mobility session. Moreover, these MIPv6-based approaches impose that both end-systems have IPv6 capabilities, in order to insert in the IP datagrams some extension headers that transport the MD's identifier (i.e., the home address) and the current MD address.

Differently, network-based mobility architectures such as Proxy MIPv6 (PMIP) [21] do not require that the MD actively participates to the mobility support signaling. Usually, these solutions specifically work in a local domain. An entity called local mobility anchor (LMA) analogous to the Home Agent plays the role of LR. In addition, a mobility access gateway (MAG), at the edge of each access network inside the local domain, performs the mobility support signaling instead of the MD. However, if the MD leaves this domain, the established sessions break because the MN would not be anchored in the local topology anymore.

Of course, all these network-layer solutions require that the network infrastructures support IPv6 capabilities. Moreover, the MIPv6 specification does not allow the simultaneous use of the multiple MD's NIC. For each given MD, the address of one NIC is only registered to the Home Agent. In addition, as demonstrated in [22], the handover latency of a MIPv6-based solution typically results higher than the handover latency of a transport-layer solution, due to the numerous authentication messages necessary in the MIPv6 registration phase (binding update).

In the architectures located between the network and the transport layers, such as Host Identity Protocol (HIP) [23] and Location Independent Addressing for IPv6 (LIN6) [24], the location registry is a DNS-like mapping function that operates as a service outside the access networks and associates host identifiers and host locators. HIP, LIN6 and similar solutions insert an intermediate layer between the network and transport layers on both the MD and CN. However, the modification on the CN appears not eligible because it may be a fixed node that could be not interested in supporting the mobility of the MD.

The common approaches working at the transport-layer protocols, such as the datagram-oriented Datagram Congestion Control Protocol (DCCP) [25], the stream-oriented mobile Stream Control Transmission Protocol (m-SCTP) [26] and the TCP enhancement TCP-migrate are very different: each given end-system plays the role of a proactive location registry that informs the CN of its configuration changes. Unfortunately, this approach fails when both the end-systems change simultaneously their IP configuration, because the two end-systems become mutually unreachable. The transport-layer solutions

require that the applications are modified on both the MD and CN to invoke the services of the novel transport layer and to implement the suitable recovery policies. This prevents the reuse of the existing applications.

The most important session-layer mobility management architectures make use of an external Session Initiation Protocol (SIP) server. SIP [27] is a session-layer text protocol that uses a message/response handshake for signaling purposes. In particular, it is used to establish or change communication parameters such as IP addresses, protocol ports and audio/video codec between the end-systems. The SIP specification allows to add application-defined fields to the SIP messages. Unrecognized extension fields will be simply discarded. When a given user U1 is available for communications, a REGISTER SIP message is sent to a SIP server that will maintain the address of the user U1 on the node N1. To initiate a call, a user U2 on the correspondent node N2 asks the SIP server, with a INVITE message, for the address of U1 and for some communication parameters. After this exchange, the two nodes may communicate directly. A REINVITE message may be used when communication parameters (such as IP address) change. The SIP protocol allows the presence of SIP proxies that can be transparent to the application (proxy agent) or can masquerade the end systems (back-to-back agent) working as an opaque relay.

Session-layer architectures, such as Terminal Mobility Support Protocol (TMSP) [28], use an auxiliary SIP server, outside the access networks, as location registry that maps a user's URI (e.g., ghini@cs.unibo.it) to the current user's location (the IP address of the user's MD). Each MD uses a SIP user agent that sends REGISTER messages to the SIP server in order to update its current location on the SIP server and INVITE messages to establish communications with the other nodes. The session-layer solutions seem to be not efficient because when an IP reconfiguration happens they require the invocation of an external localization service. In particular, the SIP-based services introduce an additional delay. This is due to the fact that when a reconfiguration of the communication parameters happens, the MD interrupts the communication, sends a SIP signaling message to the CN and waits for the response before to resume the transmission.

The MMUSE [29] approach proposes significant differences: it requires that an auxiliary SIP server (namely, the Session Border Controller, SBC) is located at the edge of the autonomous system inside which the MD will move. This autonomous system may be composed of several subnets using heterogeneous network technologies. While the MD moves across the subnets, each subnet provides the MD with a different IP address. The SBC aggregates the functionalities of SIP and Real Time Protocol (RTP) proxy, firewall and network address translation (NAT) systems, and intercepts the communications that enter and leave the network, in particular the SIP messages between the MD and its CN outside the network edge. Based on the outgoing SIP messages, the SBC sets the firewall rules to allow the subsequent SIP, RTP and Real Time Control Protocol (RTCP) communications. Moreover, when the MD moves to a different subnet and changes its IP address, the SBC modifies the outgoing datagram in order to hide to the CN the current location of the MD.

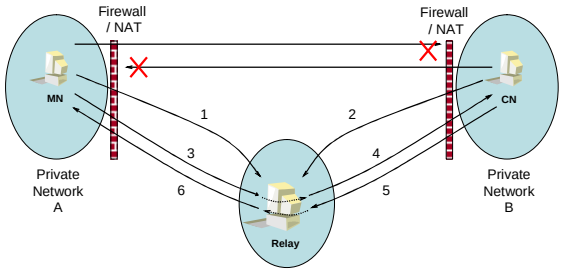

Figure 2: The data relay allows the undirected communication between two nodes behind different firewalls or NAT systems.

An important consideration is that, generally, all the described mobility management solutions do not take into account the possible presence of firewalls and NAT systems, and rely on external STUN (Simple Traversal of UDP Through Network Address Translators) [30] or TURN (Traversal Using Relay NAT) [31] servers (see Figure 2). The main related problem is that this solution introduces additional communication latencies. As we will show in the next Sections, the use of proxies in our system creates a tunnel that easily surmounts this problem.

Finally, it is worth to point out that all the described solutions do not enable the simultaneous utilization of all the NICs available on the MD. Conversely, our approach is able to simultaneously utilizing multiple wireless networks. Therefore, besides allowing a seamless communication, our approach balances also the network traffic over multiple wireless networks.

3. The Architecture

Our architecture is composed of three main modules (see Figure 3), i.e., i) a Web 2.0 mashup which offers the Web functionalities to query, discover points of interest and show them on a map, ii) a geo-localization

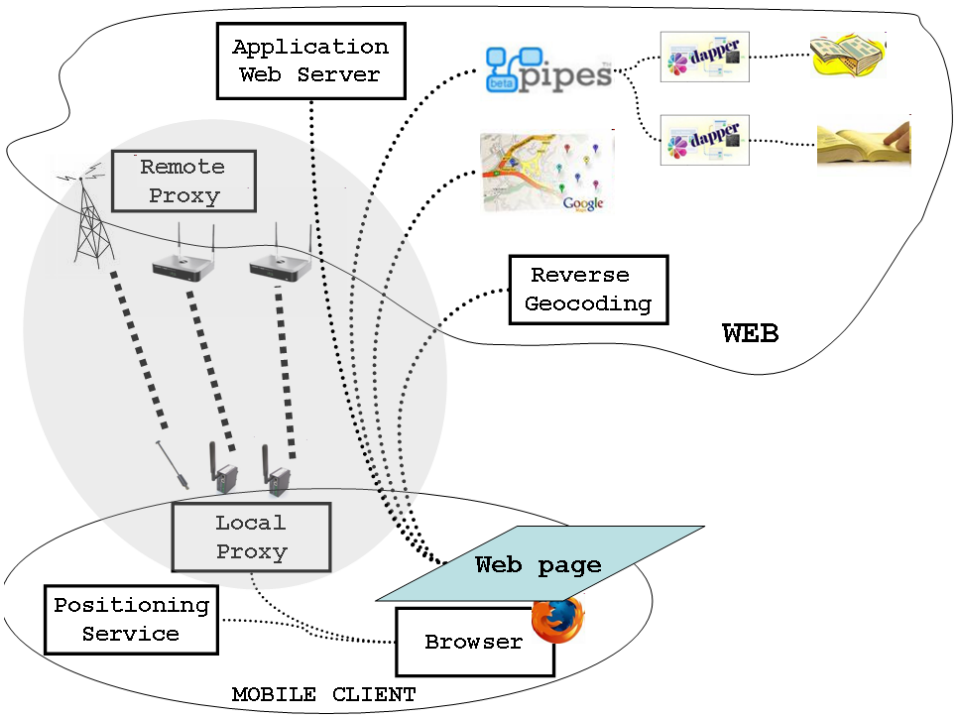

Figure 3: System Architecture

scheme, iii) a cross-layer networking service which offers a seamless communication between the mobile client and the rest of the world.

A Web service has been built that acts as the orchestrator of the mashup. As mentioned, our application is based on Web technologies, and runs on a browser. Javascript is thus used as the language to develop procedures to be run on the browser.

To enable the localization of the user, different existing options are possible. Browser plugins have been developed which allow to interact with GPSs, or Wi-Fi interfaces, in order to retrieve the position of the user. An examples is the Geode plugin recently developed at Mozilla Labs [32]. Instead, Loki is the Skyhook's browser plugin that uses Wi-Fi positioning system to enable 3rd party websites to integrate auto-location using javascript APIs. Other approaches are the Google APIs developed for Android which allow to access the data returned by a built-in GPS receiver, the system presented in [33], or more simply the use of some local Web service installed on the client terminal (if the user agrees), in charge of interacting with the GPS device and able to export such data. Alternatively, if the user prefers not to install such a kind of services on the client machine, he/she will be asked to insert his/her location manually.

A local proxy is installed on the client device, in charge of communicating with a remote proxy for ensuring a seamless navigation. It exploits several wireless network interfaces and a cross-layer networking scheme which guarantees communication even in front of horizontal or vertical handoffs.

Based on these modules, the interaction of the user is as follows. (The whole interaction flow among the distributed system modules of our architecture is depicted in Figure 4. In that Figure, MD represents the mobile device, WS represents the Web Server, RG represents the Reverse Geocoding service, Gmaps stands for Google Maps, while YP_i are the contacted yellopages-like sites.)

1. The client activates the local proxy, which connects to the remote proxy.
2. If some positioning system is available on the mobile terminal, this module should be ready for execution at the client.
3. Once the user wants to discover some points of interest, he connects to the Web application we developed (i.e., interaction between MD and WS in Figure 4). The communication and the whole interaction is performed using the cross-layer networking scheme as the underlying communication service.
4. Through the Web page retrieved at the application Web service, the user can select the typology of

places he is interested in (e.g., pubs, music shops, museums).

5. Once the selection is made, a Javascript procedure verifies if a local positioning mechanism can be utilized. If it is available, the geographical position of the user is retrieved; otherwise, the user is asked to insert such position manually.
6. If GPS coordinates are returned by the local positioning system, a reverse geocoding service [15] is queried to identify the address (i.e., street, city, country) corresponding to the geographical position (i.e., interaction between MD and RG in Figure 4).
7. Based on the type of points of interest and on the current location of the user, a mashup is activated that queries different Web services and analyzes the obtained results, which are finally shown to the user in a map, taken from Google Maps [14]. (In Figure 4, more details are depicted which describe how the mashup works; a related and more detailed description is provided in the next Section.)
8. Upon selection of a specific item by the user, a path from the current position to the selected point is shown (again, using the Google Maps service).

The following sections are devoted to present in deeper detail the Web 2.0 mashup and the cross-layer networking system, which represent the main technical novelties of our proposal.

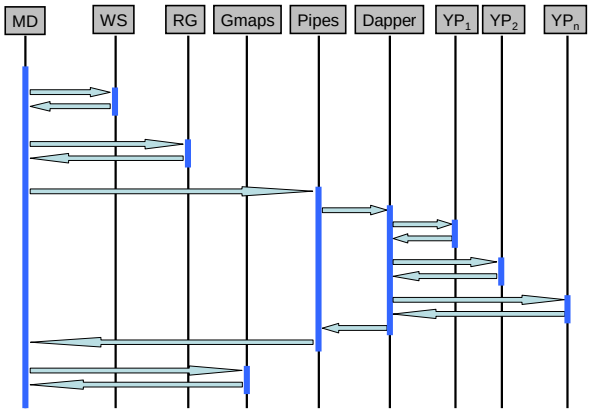

Figure 4: Interactions Flow, Application Level

4. The Web 2.0 Mashup

The developed Web 2.0 mashup module exploits HTML, Javascript RSS and JSON technologies to let the distributed entities interoperate and exchange data on the points of interests sought by the user. Four services to create the mashup have been utilized, i.e., Dapper, Yahoo Pipes, Google Maps, and a reverse geo-coding service. As mentioned, services are orchestrated by the javascript procedures implemented within the Web page retrieved by the client, and run by the browser, as shown in Figure 3.

The reverse geo-coding service is simply exploited to perform a conversion from the geographical position of the user (encoded as GPS coordinates) to a common address composed of street, city, country. This service is locally called by the browser through a Javascript procedure in the Web page retrieved by the client.

Dapper is a Web-based service that enables to extract information from any Web site and create an interactive feed from it. Data can be extracted by creating a Dapp, i.e., a Web site needs to be selected, types of data in the Web pages that should be extracted needs to be specified, together with its output format. Once properly instructed on a specific Web site, the Dapp extracts data based on the site's HTML structure. This allows to automatically query a yellowpages site and export data based on different encoding formats, e.g., RSS, XML, JSON. The automatic query can be done by building a proper URL containing the parameters of the query.

Take, for instance, a template of a query passed to the server of the Italian yellowpages (i.e., `www.paginegialle.it`). Such query corresponds to an URL such as

`http://www.paginegialle.it/pgol/4-`***`[TYPOLOGY]`***`/3-`***`[CITY]`***`?ind=`***`[ADDRESS]`***

where ***`[TYPOLOGY]`*** corresponds to the type of point of interest, ***`[CITY]`*** corresponds to the city and ***`[ADDRESS]`*** to the address where the user is currently located. This means that if one wants to look for a pub in Bologna, near the address "via Rizzoli", it is sufficient to pass to the Dapp a URL which looks like

`http://www.paginegialle.it/pgol/4-`***`pub`***`/3-`***`Bologna`***`?ind=`***`Rizzoli`***.

Then the Dapp will analyze the Web page returned by the server `www.paginegialle.it`, extracts data and encode them as an RSS feed.

In our prototype, we created two different Dapps, i.e., one to query the Italian version of the yellowpages site (`www.paginegialle.it`) , and another which queries a popular Web site to find night clubs and entertainment sites (`www.2night.it`). Both Dapps export the list of items as RSS feeds, which are passed to the next mashup module, created using Yahoo Pipes.

Yahoo Pipes is Web 2.0 service which allows to create a mashup of services by composing them as one or more pipes. During the development of the pipe, a graphical interface is provided to orchestrate these

different services. By exploiting it, a mashup has been created to call different Dapps, retrieve output RSSs coming from them, filter and merge these feeds into a unique list, and finally export them as a JSON-encoded document which is passed to the client. Each element of the list is a point of interest with its related address.

Based on this list, the Javascript executed at the browser creates a map (using Google Maps) with all points of interest displayed in it.

5. The Cross-Layer Architecture

The cross-layer networking service (depicted in Figure 5) offers a seamless communication. It is composed of two main entities, a local proxy installed on the MD and a remote proxy, installed in a fixed host, with a static IP address. The local proxy is composed of a Load Balancer Client (LBC) and of a Monitor, that dynamically configures the wireless network interfaces of the MD by selecting the best access points. The remote proxy is composed of a Load Balancer Server (LBS) and a HTTP proxy.

The Monitor and the two load balancers are general-purpose components we have developed (in a context completely different from that described in this paper). An accurate and deep description of the design and implementation of these components can be found in our previous works, in particular the architecture of both Monitor and Load Balancers is described in [36], the latest release of the load balancer algorithm is evaluated in [35] and the mechanism dedicated to the bandwidth estimation of the Wi-Fi access points is detailed in [37]. In this context, instead, we aim in highlighting the interactions among these components and the overall architecture.

Moreover, it is important to point out that the seamless communication service provided by the cross-layer architecture operates as a separate session-layer service implemented in the two end-systems only, the MD and the fixed host. The service operates on top of (and as a complement of) the existing IP-based network infrastructures. Thus, it can coexist with other architectures such as, for instance, 3GPP.

Basically, the service maintains the abstraction of an always available TCP connection between a Web browser in the MD and the HTTP proxy in the remote fixed host, despite of handoffs and changes of MD IP addresses. The service also balances the load among the different MD network adapters and recovers connection faults, but using the long-range communication technology only when strictly necessary (i.e., when no Wi-Fi APs are available).

The Web browser is simply configured to use the LBC as its proxy, so as to deliver/receive the HTTP request/response to/from the LBC through a TCP connection. For each connection with the Web browser, the LBC maintains a session-layer multi-path communication channel (MPCC), through which the HTTP messages are transmitted in both directions, as illustrated in Figure 4. This session-layer MPCC consists of a dynamic set of Secure Socket Layer (SSL) [34] connections, one for each working MD wireless adapter, that guarantee the privacy of the transmitted data. Each SSL connection is supported by an exclusive TCP connection, which binds its local IP address to the address of one of the working MD wireless network adapters, and sends and receives TCP segments through this wireless adapter only. For each MPCC between the LBC and the LBS, the LBS maintains a TCP connection with the HTTP proxy.

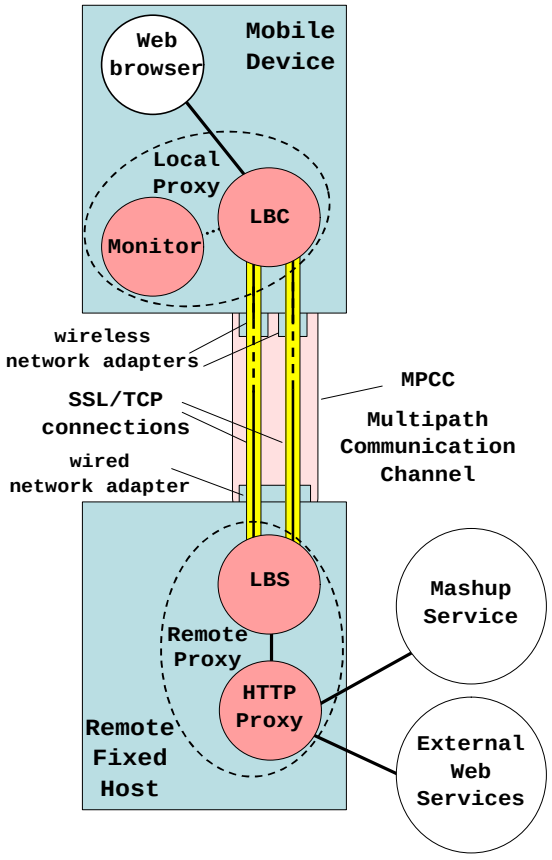

Figure 5: Architecture for Seamless Navigation

The HTTP requests produced by the Web browser reach the LBC as a TCP flow; the LBC splits that flow and delivers the pieces through the MPCC, retransmitting the pieces when necessary. The algorithm to balance the network traffic is described in full details in [35]. In substance, the load balancing algorithm estimates the performance of each MPCC's TCP connection bound to each NIC. The

LBS receives the pieces, reconstructs the original TCP flow and delivers it to the HTTP proxy that forwards the request to the destination. Collected HTTP responses follow the same path but in the opposite direction.

The Monitor at the MD is responsible for the dynamic configuration of the datalink, network and session layers of the MD (see [36] and the next subsection for more details), in particular for the Wi-Fi NIC. Using cross-layer information, the Monitor detects when a NIC leaves the coverage area of the associated AP. In this case, the Monitor informs the LBC that closes the SSL/TCP connection (of the MPCC) that uses that NIC. Then, the Monitor drives the NIC in scanning the Wi-Fi channels to detect the available APs and to estimate the traffic they are subject to, so as to exclude the APs that provide marginal signal strength. Following a model described in [37] that take into account both the traffic and the signal strength, the Monitor estimates the bandwidth each AP may provide to the NIC, excludes those APs that are associated with the other NICs of the MD itself, and selects the AP that provides the higher bandwidth (the interested reader may refer to [37] for a detailed description of the model). Finally, the Monitor associates the NIC to the selected AP, configures the IP address and the routing rules for the NIC, and informs the LBC of the newly available NIC. The LBC creates a new SSL/TCP socket bound to that NIC, connects the socket to the LBS, and inserts in the session-layer MPCC the new SSL/TCP connection between LBC and LBS.

The described procedure guarantees that the MPCC always maintains and uses, for load balancing purposes, an SSL/TCP connection for each working wireless network adapter of the MD.

All the described software entities (LBC, LBS, HTTP proxy) have been implemented at the application layer and do not need particular operating system features. The only exception is the implementation of the Monitor, that is composed of several separates applications, and relies on some features and APIs of the GNU Linux operating system, in particular those that enable the communication with (and the configuration of) the datalink and network layers of the MD. For instance, the Linux Wireless Extensions API for the traffic analysis and the datagram-oriented Netlink socket to manage the dynamic routing tables [38].

The next subsection will describe in more detail how the monitor works.

5.1. Monitor Implementation

The MD is governed by the Linux operating system, and its software has been designed, for backward compatibility purposes, so as to allow one to use old-style Linux kernels, such as those of the versions 2.4.x (the most recent version is 2.6.30, at the time of this writing). This backward compatibility enables the implementation of the MD even using PC boards and handheld devices running old Linux kernel versions. Moreover, the Linux kernel can be configured so as to manage multiple routing tables at the network layer. Such a configuration allows this kernel to perform dynamic routing of IP packets based not only on the packet destination address but also on its source address. Note that one such a configuration can be generated easily by simply compiling the Linux kernel source code with a specific flag appropriately set in the makefile.

The software running in the MD is structured as depicted in Figure 6. The shaded boxes in this Figure represent the MD software components that we have designed. At the application layer, the LBC receives the data flow through an HTTP/TCP connection with the Web browser and delivers it to the LBS through the MPCC.

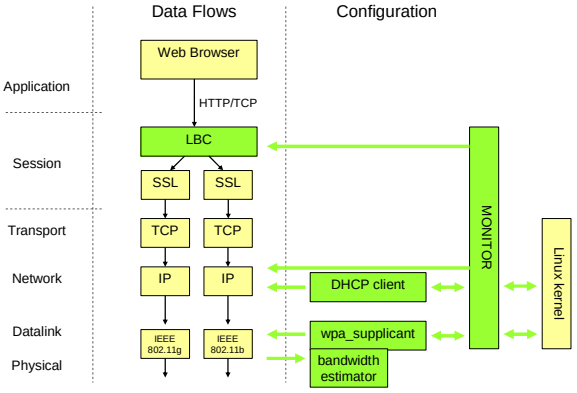

Figure 6: Mobile Device Software Architecture

The Monitor communicates with the Linux kernel by means of a datagram-oriented Netlink socket: when a network adapter changes its status, the kernel informs the Monitor, which in turn activates the adapter reconfiguration procedures. Moreover, the Monitor communicates with the session-layer LBC (see below) to inform it that a wireless network adapter has been completely configured and can be used.

The lowest software layer, i.e., the data link layer, implements the operations that allow each wireless network adapter in the MD to detect the presence of an access point and to configure its own data link layer so as to communicate with that access point. In addition, the data link layer is responsible for managing the network adapter that looses the access point carrier:

that adapter cannot communicate and thus must be disabled.

All these operations have been implemented as a separate process, reusing a widely used, free and open-source application called wpa_supplicant. Wpa_supplicant is not dependent on the wireless technologies being deployed as it makes use of services provided by the wireless adapter driver. For this reason, the use of wpa_supplicant allows the MD to accommodate network adapters of wireless technologies different from IEEE802.11.

As concerns authentication and security of wireless networks, wpa_supplicant encloses all functionalities able to cope with them. As to the authentication functionality, it requires the use of a configuration file in the MD, which contains the information necessary to enable wpa_supplicant authentication in each visited network. This information depends on the kind of authentication a visited network requires; for instance, WEP is the simplest (and less secure) system; it is based on the knowledge of a key shared among all users. The access point asks wpa_supplicant for the name of the wireless network (the so-called Extended Service Set Identifier – ESSID) and the shared 128 bit WEP key. In contrast, other systems require external authentication servers, such as those based on Robust Security Network (IEEE802.11i [39]), IEEE802.1X [40], Extensible Authentication Protocol (EAP) [41] and Remote Authentication Dial In User Service (Radius) [42].

Obviously, the user MD may access only to the access networks of which the user possesses the necessary permissions. The user is responsible to acquire the required authentication information from each access network's administrator and to write the configuration file required by the wpa_supplicant.

As to the security functionality, wpa_supplicant hides the complexity of different security mechanisms. Simply stated, in the IEEE 802.11 scenario, wpa_supplicant drives the wireless adapter in scanning the radio channels to detect the access points of the wireless networks of which it knows the authentication information. Actually, when more access points are available, the original wpa_supplicant selects the one that provides the best radio signal, and executes the authentication procedure with that access point. Hence, we have implemented two minor modifications of the wpa_supplicant access point selection process. The first one consists in a preliminary bandwidth estimation phase to identify the access point that is able to provide the MD with the higher bandwidth. This phase is implemented using the Linux Wireless extensions API and is based on a initial traffic monitoring and on a traffic model [35] that take into account traffic, error rate and signal strength. The second modification ensures that additional network adapters of a given MD select unique access points to provide the MD with multiple independent wireless paths.

The network layer implements those operations that configure the IP address, the netmask and the gateway IP address of the wireless network adapters, and sets up the rules for the dynamic routing of the IP datagram via those adapters. These operations are executed after wpa_supplicant has configured the data link layer of a wireless adapter, by associating it to a given access point. In fact, the kernel informs the Monitor as soon as one such an association is established; then the Monitor starts a Dynamic Host Configuration Protocol (DHCP) client. The DHCP client requests an IP address, a netmask and an IP gateway to the DHCP server of the visited network. When the DHCP client terminates successfully the configuration of the network layer adapter, the Monitor sets up a new rule and a new routing table (invoking the "ip rule" and "ip route" commands) in order to enable the dynamic routing via this adapter. These rule and routing table impose that the IP datagrams having the same source IP address as this adapter be routed through it, regardless of their IP destination address. As this adapter is configured, it can be used by the session layer. In this way, the IP datagram of the TCP connection bound, by the LBC, to a given adapter are routed and transmitted through that adapter.

When the network adapter looses the access point carrier, the adapter's driver interrupts the kernel. This contacts the Monitor to disable that adapter and delete the routing rule for this adapter. Hence, the adapter will be unavailable until it is reconfigured.

6. Experimental Evaluation

In this section we study the behaviour of our developed application, based on a qualitative and quantitative evaluation. Hence, first, we show some screenshoots obtained as a result of the use of the application. Second, we report on measurements we obtained to access the system when the user passes through different networks coverage.

Figure 7 shows the result of a query process. In particular, the position of the user is shown with the flag in the map, thank to the use of Google Maps. The block reported below the image shows a list of points of interest, which are located near the user, found thank to the mashup. Figure 8 shows the result obtained when the user selects one of the items reported in the list. The path to reach the interest point is shown.

The objective of the experimental evaluation was to assess the ability of our architecture to both guarantee the continuity of the communication and deliver the application data within an acceptable level of

interactivity, in spite of MD movements and changes of access networks. To this end, the experiments concerned the performances of the MPCC that connect a given MD with the Remote Fixed Host (RFH), in particular the elapsed time necessary to deliver, from the MD to the RFH, each application frame packet (1126 Bytes) with a frequency of 25 frames/second.

Figure 7: A screenshot of the result of a query

Figure 8: Path to reach the selected point of interest

The experimental scenario was as follows. A human operator, carrying an MD, has covered a route of 80 meters, as depicted in Figure 9. The MD is equipped with two IEEE 802.11g wireless interfaces. Along that route two IEEE 802.11g access points were available, indicated with AP1 and AP2 in Figure 9. The access point AP1 worked on channel 1, and was located near the beginning of the MD path, whereas the access point AP2 worked on the channel 6 and was located close to the end of that path. The transmission signal of these two APs was strongly and abruptly obscured by walls. Figure 10a illustrates the signal power received from the two APs along the path, due to the obstacles; it is worth to point out that below a threshold of -80dBm the AP becomes practically unavailable. In other words, the access point AP2 was unavailable at the beginning of the MD path whereas the access point AP1 was unavailable at the end of the path.

Moreover, both the access points were affected by an additional background traffic, of approximately 5Mbps, generated by two others static devices, and there were other three APs to cause inter-channel interferences.

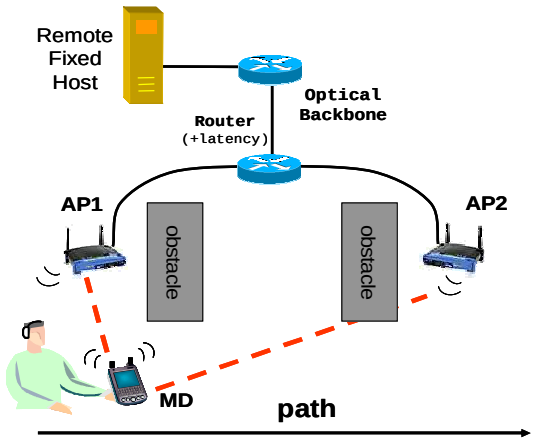

Figure 9: Experimental scenario

The two APs (AP1 and AP2) were connected to the RFH through a 1000Mbps Gigabit Ethernet network and an optical fibre backbone, and a Linux-based router was set to introduce an artificial latency of 50 ms, with a tolerance of 7%. This choice allows us to analyze the system behaviour in a realistic scenario.

The frame transmission has been carried out, in the following three modes, while moving along the described path: in the first two modes we have used the access point AP1 and access point AP2 in isolation, respectively, in order to show the performance of each single access point along the route of our experimental scenario. In the third mode, our system exploits both the access points for the frame transmission.

Figures 10b, 10c and 10d illustrate the frame delivery time we have obtained using AP1, AP2 and both together with our system, respectively.

In these three figures the x axis represents the frames sent from the MD while the y axis is the delivery time of each given frame. Close to the end of the MD path, the access point AP1 firstly introduces considerable delays and then it becomes unavailable, thus the frames do not reach the RFH and there are no delivery times, as

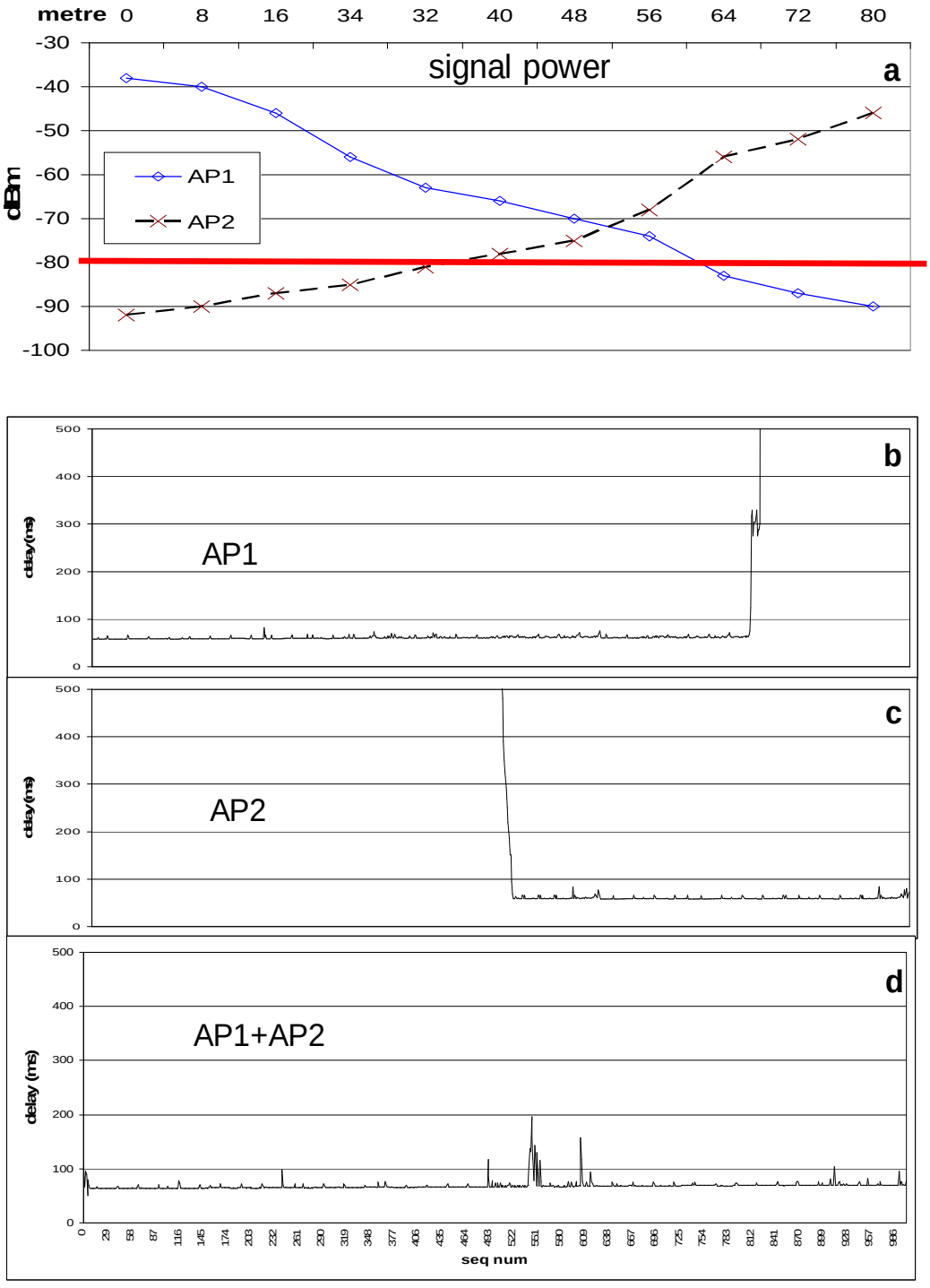

Figure 10: Experimental scenario and measures

illustrated in Figure 10b.

On the contrary, the access point AP2 (see Figure 10b) is unavailable at the beginning of the MD path and introduces small delays at the end of the route. Instead, our proposed system selects the suitable access point along the overall MS route, delivering all the frames to the destination. As depicted in Figure 10d, due to the retransmission, there are some peaks of the frame delivery time, but almost all the frames do not overcome the threshold of 150 ms, and only few (2 over 1000 frames) reach the destination after 150 ms but before 200 ms. Our system introduces just a little overhead: in fact, the amount of bytes retransmitted is small, i.e. 23 frames on a total of 1,000 frames. These results confirm that the proposed system provides responsiveness, reliability, continuity of the communications, and guarantees the viability of services offered by our mashup service.

It is worth to point out that performances do not worsen even when a number *n* (*n*>2) of access points are available in the route. In fact, when a given NIC loses the wireless carrier, the Monitor uses the bandwidth estimator described in [37] to estimate the bandwidth of each available APs. Then, it associates that NIC to the AP with the highest available bandwidth. The same procedure is executed every time the NIC leaves the coverage area of its associated AP.

Finally, the heterogeneity of the NICs on the MD does not represent a difficulty. In fact, the two load balancers abstract from the underlying network technologies and treat each communication passing through a specific NIC as a standard TCP communication channel between them. Moreover, the two load balancers estimate the performances of each TCP connection in terms of latency and bandwidth. This allows to balance the traffic workload among all the available NIC independently of the NIC technologies.

7. Conclusions

We have presented a system for a location-based discovery of points of interest. We have demonstrated that the concurrent use of Web 2.0 technologies and cross-layer networking schemes can guarantee a seamless use of sophisticated Web services, while moving through different and heterogeneous wireless networks.

The key for offering a seamless interaction among the mobile terminal and the Web services deployed over the Internet is the use of a couple of proxies. One is locally installed on the mobile terminal, while the other is made available on the Internet to all wireless devices. This solution enables wireless devices to exploit different networking technologies while being involved in a single session. The experimental results we obtained confirm such claim.

8. References

[1] S. Ferretti, V. Ghini, "A Web 2.0, Location-Based Architecture for a Seamless Discovery of Points of Interests", Telecommunications, 2009. AICT '09. Fifth Advanced International Conference on , pp.226-231, 24-28 May 2009.

[2] R. Lin, Y. Zhao, H. Zou, F. Yang, "An Enhanced Location Service with Social Feature", Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on 12-14 Oct. 2008, 1 - 4.

[3] J.C. Kim, T.W. Heo, J.W. Kim, J.H. Park, "Ubiquitous location based service", Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE, Sept. 2005, 841 - 845.

[4] D. Mohapatra. S.B. Suma, "Survey of location based wireless services", Personal Wireless Communications, 2005. ICPWC 2005. 2005 IEEE International Conference on 23-25 Jan. 2005 358 - 362.

[5] C.T. Wu, H. Mei, "Location-based-service roaming based on Web services", Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on, March 2005, Volume: 2, 277- 280.

[6] P. Bellavista, A. Küpper, S. Helal, "Location-Based Services: Back to the Future", *IEEE Pervasive Computing* 7, 2 (Apr. 2008), 85-89.

[7] Google Mobile, November 2008, http://www.google.com/mobile/index.html.

[8] Around me application, http://www.tweakersoft.com/mobile/aroundme.html, 2008.

[9] Skyhook Wireless Web Site: http://www.skyhookwireless.com/

[10] "Skyhook powers Apple's new location apps", The Journal of New England Technology, January 2008, (last checked, July 2009), available at: http://www.masshightech.com/stories/2008/01/14/daily20-Skyhook-powers-Apples-new-location-apps.html

[11] S. Ferretti, P. Salomoni, M. Roccetti, S. Mirri & L.A. Muratori, "At the Crossroads of Web and Interactive Multimedia: an Approach to Merge the Two Realms", Proc. CCNC 2009, Las Vegas (USA), IEEE, January 2009.

[12] Dapper, November 2008, http://www.dapper.net/.

[13] Yahoo Pipes, November 2008, http://pipes.yahoo.com/pipes/

[14] Google Maps, November 2008, http://maps.google.com/

[15] Reverse Geocoding Webservice, November 2008, http://www.geonames.org/export/reverse-geocoding.html

[16] GeoRSS Web site: http://georss.org/

[17] E. Gustafsson and A. Jonsson, "Always Best Connected", IEEE Comm. Mag., vol. 10, no. 1, Feb. 2003, pp. 49–55.

[18] D. Johnson, C. Perkins, J. Arkko, "Mobility support in IPv6", RFC 3775, June 2004.

[19] R. Koodli, "Fast Handover for Mobile IPv6", IETF RFC 4068, July 2005.

[20] H. Soliman, C. Castelluccia, K. ElMalki, L. Bellier, “Hierachical Mobile IPv6 (HMIPv6) Mobility Management”, IETF RFC 5380, Oct. 2008.
[21] S. Gundavelli, V. Devarapalli, K. Chowdhury, B. Patil, “Proxy Mobile IPv6”, RFC 5213, August 2008.
[22] Dong Phil Kim, Seok Joo Koh, "Analysis of Handover Latency for Mobile IPv6 and mSCTP", Communications Workshops, 2008. ICC Workshops '08. IEEE International Conference on , vol., no., pp.420-424, 19-23 May 2008.
[23] R. Moskowitz, P. Nikander, “Host Identity Protocol (HIP) Architecture,” IETF RFC 4423, May 2006.
[24] F. Teraoka, M. Ishiyama , M. Kunishi ,“LIN6: A Solution to Multihoming and Mobility in IPv6”, IETF Internet Draft, draft-teraoka-multi6-lin6-00.txt, Dec. 2003.
[25] E. Kooler et al., “Datagram Congestion Control Protocol (DCCP)”, IETF RFC 4340, March 2006.
[26] M. Riegel, M. Tuexen, “Mobile SCTP,” IETF Internet draft, draft-riegel-tuexen-mobile-sctp-09.txt, Nov. 2007.
[27] J. Rosemberg et al., “SIP: session initiation protocol”, IETF RFC 3261, June 2002.
[28] Teck Meng Lim, Chai Kiat Yeo, Francis Bu Sung Lee, Quang Vinh Le, "TMSP: Terminal Mobility Support Protocol," IEEE Transactions on Mobile Computing, vol. 8, no. 6, pp. 849-863, June 2009.
[29] S. Salsano et al., “SIP-based mobility management in next generation networks”, IEEE Wireless Communications, pp. 92-99, April 2008.
[30] J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, “STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)”, RFC 3489, March 2003.
[31] J. Rosenberg, R. Mahy, C. Huitema, “Traversal Using Relay NAT (TURN)”, Internet-Draft, draft-rosenberg-midcom-turn-08, September 2005.
[32] Mozilla Labs, Geode, http://labs.mozilla.com/2008/10/introducing-geode/.
[33] D. Carboni, A. Piras, S. Sanna, S. Giroux, “The web around the corner: augmenting the browser with gps”, in *Proceedings of the 13th international World Wide Web Conference on Alternate Track Papers &Amp; Posters* (New York, NY, USA, May 19 - 21, 2004). WWW Alt. '04. ACM, New York, NY, 318-319.
[34] A.O. Freier, P. Karlton, P.C. Kocher, "The SSL protocol version 3.0" Internet draft, draft-ietf-tls-ssl-version3-00.txt, November 1996.
[35] V. Ghini, G. Lodi, S. Cacciaguerra, F. Panzieri, "Meeting Interactivity Requirements in Mobile E-Witness: an Experimental Study", Wireless Personal Communications, Springer, accepted for publication, July 2008. Available online at http://dx.doi.org/10.1007/s11277-008-9630-y
[36] V. Ghini, G. Lodi, F. Panzieri, “Mobile E-Witness”, Multimedia Tools and Applications (Springer), vol. 37, num. 3, pp.293-318, May 2008.
[37] V. Ghini, S. Cacciaguerra, G. Lodi, F. Panzieri, “Enhancing Mobile E-Witness with Access Point Selection Policies”, Proc. of IEEE Information Technology: New Generations (ITNG 2008), Las Vegas, NV, USA, April 2008.
[38] Linux wireless extensions api. http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html. 2008.
[39] IEEE Std. 802.11i-2004, “Medium Access Control (MAC) Security Enhancements,” July, 2004.
[40] IEEE Std. 802.1X-2001, “IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control,” June, 2001.
[41] B. Aboba et al., “Extensible Authentication Protocol (EAP)”, RFC 3748, 2004.
[42] B. Aboba, P. Calhoun, “RADIUS (Remote Authentication Dial In User Service) support for Extensible Authentication Protocol (EAP)”, RFC 3579, September 2003.