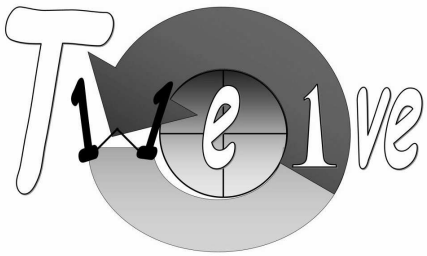
UNIVERSITÀ DEGLI STUDI DI TRENTO

DIPARTIMENTO DI INFORMATICA E TELECOMUNICAZIONI



TWELVE PROJECT[1]

Uni-Fy: Code Overview and Product Manual

Mauro Brunato, Renato Lo Cigno

Danilo Severina

Version 1.3

October 2006

[1]The TWELVE Project is a PRIN project and it is supported by Italian Ministry for University and Research (MIUR)

Contents

Chapter 1

System overview

The Uni-Fy system implements a wireless open access network gateway. Its main features are:

- support for multiple external authentication providers
- extensible support for several authentication techniques
- firewalling
- accounting

Its modular design is intended for easy implementation of novel features such as QoS control, context-dependent services and traffic shaping and modeling.

The system is fully implemented in C++ as a collection of user-space applications. Linux was chosen as operating system, but portability to other operating systems, in particular Windows, is one of the main criteria in development; however, linux-specific optimizations, such as the implementation as kernel-space modules, are being considered.

Installation and configuration instructions are available in Chapter 5. Advanced features such as Virtual LAN management and Network Address Translation are also available via Linux kernel as described in Chapter 7.

1.1 Basic network configuration

Figure 1.1 shows an overall view of the proposed network components.

- The blocks "**AP**" denote wireless access points.
- The **Gateway** component is basically a layer-3 switch with some additional ad-hoc functionalities (however, a configurable router or a firewall with enough flexibility should be applicable): it checks packets that are directed from and to authorized clients and puts them into the right interface.
- The **Gatekeeper** component receives unauthorized packets from the Gateway: packets are subjected to processing and are used to trigger events such as an authentication procedure. The authentication procedure will involve the client, the Gateway (for packet forwarding), the Gatekeeper and an authentication provider,

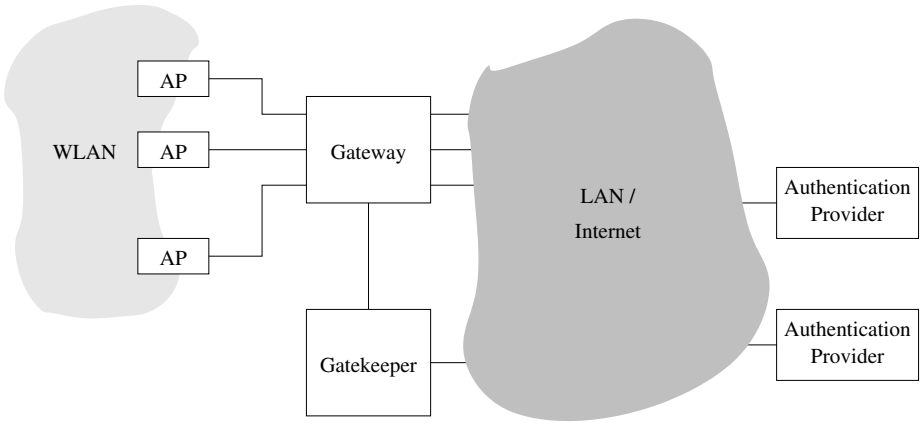


Figure 1.1: An overall view of the Uni-Fy system.

possibly chosen among many. The role of every component and the type of interaction depend on the authentication mechanism, and shall be discussed later.

For sake of clarity, connections among components are shown as separate wires in Figure 1.1. However, many logical connections can be shared by separate virtual LANs on the same wire; the Gateway and Gatekeeper components can even reside in the same machine. If the AP supports an evolute operating system, the whole solution could be collapsed into a single piece of hardware: however, high performance requirements may force physical separation of the components.

The Gateway component bridges the wireless LAN and the other side, which may be a corporate LAN, a WAN, a point-to-point line with several network components in it. Non-interference with other network components is an important requirement.

Chapter 2

Block structure

Figure 2.1 shows the basic block structure of the Uni-Fy system. It consists of two main blocks, the *Gateway* and the *Gatekeeper*; their detailed description follows.

2.1 The Gateway component

Figure 2.2 shows the detailed block structure of the Gateway for the Uni-Fy system.

The Gateway machine contains various network interfaces, some of which are directed to the wireless LAN, some to the wired LAN. Packets to and from the interfaces are managed by the *router* module.

The router module identifies packet sources and destinations according to their Ethernet MAC address and their IP address. Correspondence between MAC and IP addresses for authorized clients is stored in the *client list*, accessed by the router via a read-only interface.

The router takes decisions on packets based on the client authorization status, as explained above, and on the packet protocol (at network and transport level); rules governing the router's behavior are stored in the *Rule Table*, created at startup and accessed by the Router via a read-only interface.

In addition to the physical LAN interfaces, the router can also send and receive packets via a UDP port connected to the Gatekeeper. In principle, all unauthorized packets are sent to the Gatekeeper for further processing, while all packets received from the Gatekeeper are forwarded to the appropriate interface, with no interference from the Gateway rules.

Finally, the router module can optionally encapsulate all dropped frames into UDP envelopes and send them to a specific address for debug purposes.

Rules and authorization tables are kept as simple as possible in order to avoid any bottleneck and to simplify implementation on diskless dedicated hardware with a limited amount of computational power.

The client list is updated via a "command" interface, which receives simple textual commands from the Gatekeeper.

2.2 The Gatekeeper component

Figure 2.3 shows the block structure of the Gatekeeper for the Uni-Fy system.

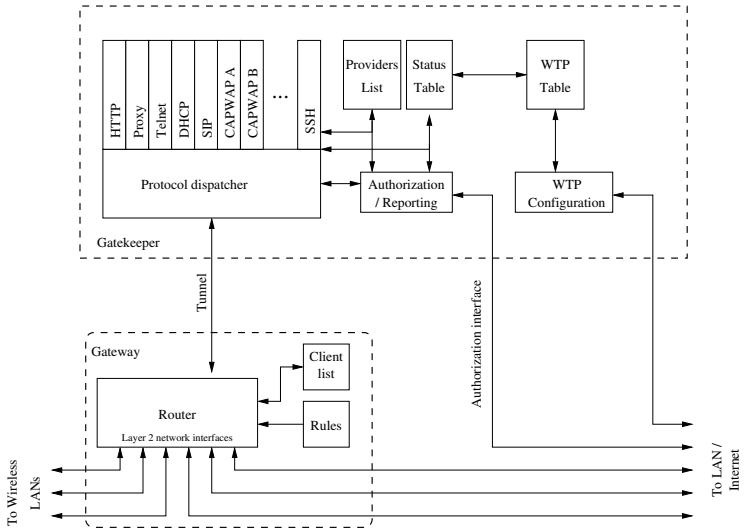


Figure 2.1: Block diagram of the Uni-Fy system

The Gatekeeper component performs all tasks that require more processing than just looking at frame and packet headers. It basically receives all unauthorized packets from the Gateway through the "tunnel" shown in Figure 2.1. This tunnel is possibly implemented as a pair of UDP sockets. In case of Uni-Fy implementation within a single machine, a bidirectional IPC channel could be considered. Packets are processed with the aim of authenticating the user, and eventually sent back to the Gateway to be forwarded, or bounced.

The main functions of the Gatekeeper are:

- Client status maintenance; the Gatekeeper's status table is essentially a superset of the Gateway client list.
- DHCP management (either DHCP server or DHCP Relay);
- Client authentication and authorization.

The Gatekeeper acts according to the packets received from the Gateway component, updates its status table and issues commands back to the Gateway component. Commands are of the following form:

- Send a packet through a specified interface;
- Authorize client identified by MAC/IP pair
- Revoke authorization to a given client.

In addition to the Gateway tunnel, Gatekeeper functionalities can be controlled via two TCP sockets, one devoted to client authorization functions, the other to status reporting.

2.2.1 The Status Table

The core of the Gatekeeper system is the client status table, shown in figure 2.4.

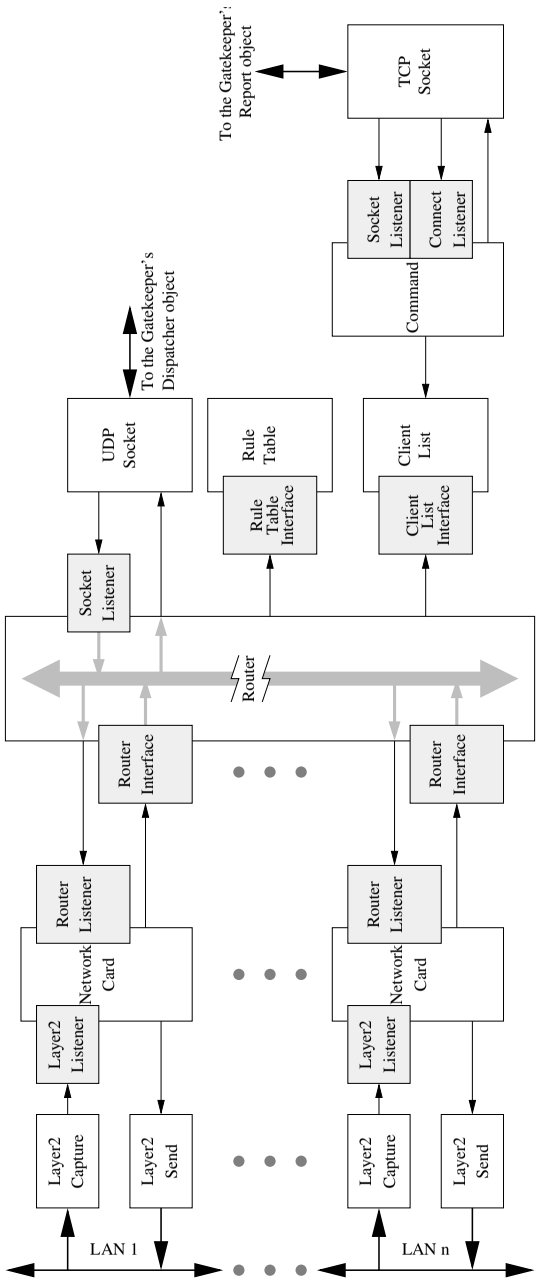


Figure 2.2: Block diagram of the Gateway component.

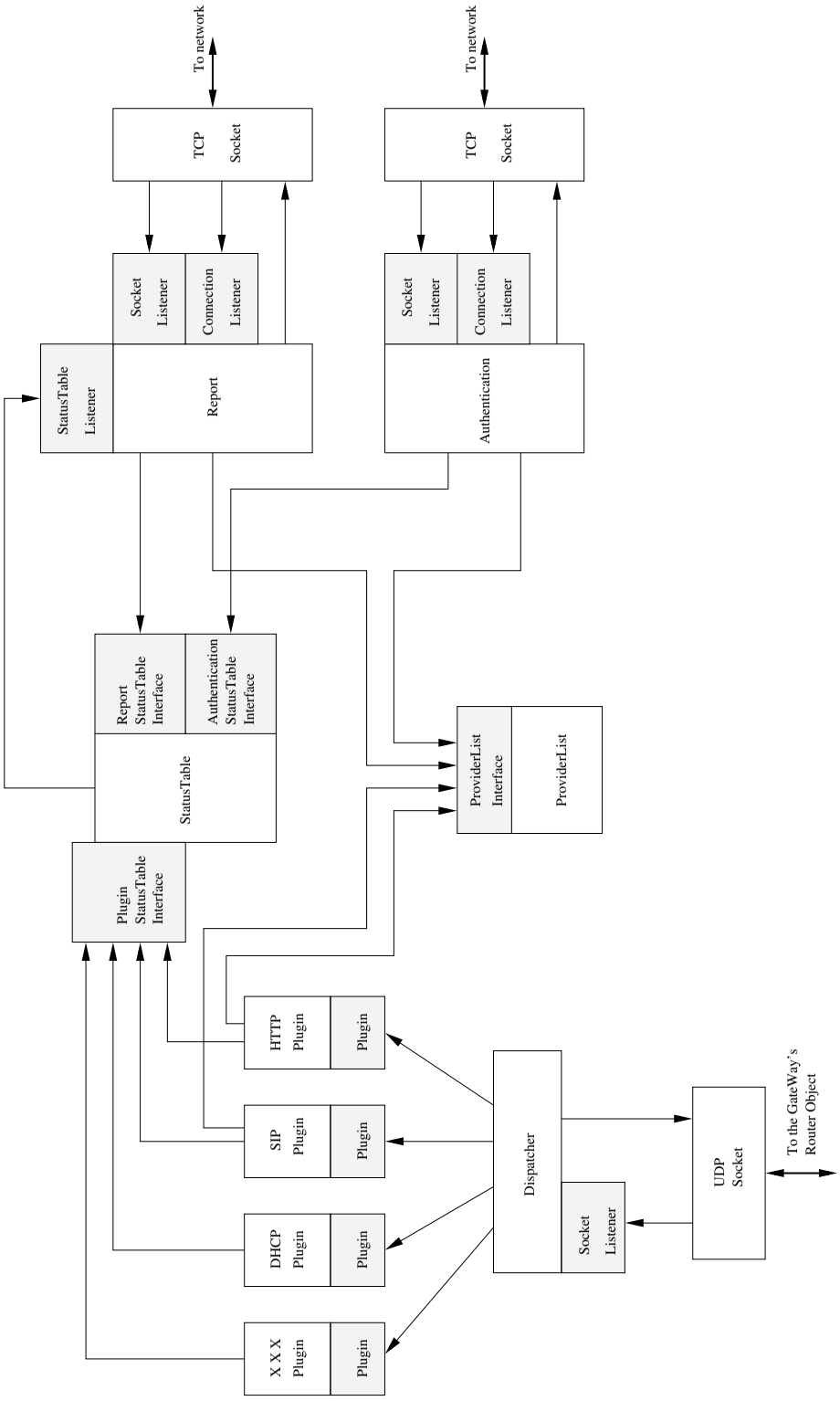


Figure 2.3: Block diagram of the Gatekeeper component.

The Status table represents the current status of the network. Every table row is associated to one IP address of the wireless subnet, and contains status information such as the Ethernet MAC address of the currently associated wireless client, traffic information, the identity of the person who got the authorization and, after the introduction of the SIP plug-in, also information about the type of authorization performed (if any).

In particular, each entry contains two meaningful status variables representing the DHCP association status of the corresponding IP address, and the type of the authorization associated with that IP (read below for more information). Figure 2.5 shows the state transition diagram managed by the system. Every IP address can be in one of six states:

FORBIDDEN The IP address is reserved for system use.

FREE The IP address is free and can be associated to a new client upon a DHCP request.

FIRST_SEEN The IP address has been offered to a new client during the discovery process.

TEMPORARY_LEASE The IP address is leased to the client with a short expiration time.

AUTHORIZED The client has been identified and gateway services are granted. Subsequent DHCP leases will confirm the current IP address with a longer expiration time.

On normal conditions, upon connection and identification, status moves clockwise in Figure 2.5 from `FREE` to `AUTHORIZED`; upon logout, the counterclockwise solid arrows from `AUTHORIZED` to `FREE` are followed. Abnormal transitions are represented by dashed arrows.

As said before, the introduction of the SIP plug-in introduced a new status variable, whose role is to keep track of which kind of authentication has been performed by a certain user. Notice that before the addition of this plug-in, only HTTP authorization were achievable, requiring the use of a web browser with javascript functionalities enabled or multi-windows capable. Thus, each IP address/user is associated with only one of the following authorization type at a time:

NONE The IP address has not yet been authenticated

UNKNOWN This value is not really needed but can be useful as a *warning message* when new types of authentication are introduced. In fact, when an authentication provider contacts the authenticator module in order to move the status of a user to `AUTHORIZED` or making a renewal of that authentication, it has to send a message indicating if it is a SIP, an HTTP, or whatever authenticator. If for some reason, a new kind of authentication is introduced, and erroneously the new authenticator try to perform a not yet implemented authentication, the value `UNKNOWN` is chosen as authentication-type value. Thus, if such a value is associated with a certain IP, then the administrator of the system knows immediately that something is not working properly.

PERSISTENT The IP address has been configured as permanently authorized in the Gatekeeper configuration file (no renewal needed)

HTTP The IP address has been moved to `AUTHORIZED` status by an HTTP authenticator

SIP The IP address has been moved to `AUTHORIZED` status by a SIP authenticator

2.2.2 The WTP Table

WTP (*Wireless Termination Points*) Table is used only if the CAPWAP plug-in is enable and it contains status and configuration of WLAN AP. In particular each entry contains essential information of CAPWAP protocol, such as Session Identifier, and the specific configuration of APs (e.g. status, radio channel and transmission power). The status of AP is made up by two fields: (*i*) AP Status and (*ii*) Configuration Status of Configuration.The description of possible states each AP follows the definitions that are provided by the CAPWAP protocol; the list of the *AP Status* follows:

IDLE AP is not active;

DISCOVERY AP is in discovery phase;

JOIN AP is in join phase;

SULKING AP is in sulking state;

CONFIGURE AP is in configure phase;

RUN AP correctly running;

RESET AP is in reset phase.

Instead the *Configuration Status* can be:

NOPRECONFIGURATION AP has not assigned configuration;

NOACTIVEPRECONFIGURATION AP is not active but it has assigned a configuration;

ACTIVEPRECONFIGURATION AP is active and has the assigned configuration;

ERRORPRECONFIGURATION error occurs in the AP configuration;

PENDINGPRECONFIGURATION AP has pending configuration.

2.2.3 The Providers List

The Providers List contains a list of authorized authentication providers that, at the moment, can be an HTTP authenticator, the native one, or a SIP one. Each provider is associated to a name, an IP address, and some data about the identification process. This list is used to recognize the IP address that can be reached by not-authenticated user: the users have not an authorization must be allowed to contact the authentication provider to receive permissions.

For example, a possible identification process is a “captive portal” authentication (see section 3.2), where a provider entry must contain the URL the client must be directed to in order to initiate the authentication process and a shared secret for mutual authentication. Another possible identification process is the SIP authentication procedure (see section 3.4), where a SIP-enabled phone/softphone (properly configured) can be authorized just switching it on and let it perform automatically all the registration steps.

2.2.4 The Dispatcher

The main component in the Gatekeeper architecture is the Dispatcher. It represents the Gatekeeper's tunnel endpoint. When a packet is received through the tunnel (meaning that Gateway requires some processing), the Dispatcher polls protocol-specialized modules, called "plug-ins" (see 2.2.5), until it finds one that can handle the packet. The selected plug-in module can forge some reply packets that are sent back by the Dispatcher via the tunnel.

2.2.5 The plug-in modules

The plug-in modules are used by the Dispatcher module in order to take decisions about specific protocols. They all expose a public method which is invoked by the Dispatcher in order to submit a packet; this function returns a response packet, if needed, and some indication about the subsequent action (e.g., send back the original packet to the Gateway for forwarding, send a new packet). If necessary, the plug-in modules can access and modify the status table.

The DHCP plug-in The DHCP plug-in module manages the DHCP service in the wireless LAN. When the Dispatcher receives a DHCP packet, it submits it to the DHCP module. To decide the appropriate action, the DHCP module can access the authorization list, where the status of every IP address is stored.

The DHCP Relay plug-in The DHCP Relay plug-in module acts as a Relay Agent for DHCP transactions. This module can be used **instead of** DHCP plug-in module. When the Dispatcher receives a DHCP packet, it submits it to the DHCP Relay module. This module forwards the packet to DHCP server in the external LAN to obtain an IP address for the client. DHCP Relay module interacts with the authorization list to update users' status.

The HTTP plug-in The HTTP handles direct HTTP connection requests that bypass the Proxy server. Direct HTTP connections might be issued in case the client has disabled the proxy service.

The HTTP module forges an entire connection session (setup handshake, HTTP GET/response and shutdown handshake) by spoofing the requested web server. Its fixed response to any client request is a redirection toward a predetermined page, usually containing instructions to correctly set the proxy server in the client system.

The Proxy plug-in The proxy module monitors communications between the unauthorized wireless clients and the proxy server of the fixed network. It is usually transparent, and it only acts when a client that is not authorized requests a web page which does not belong to an authentication provider. In this case, the Proxy module generates an HTTP 302 response that redirects the client to an authentication page. Moreover, the necessary FIN packets are forged in order to shut down the proxy communication and force the client to set it up again (otherwise sequence numbers would not match anymore).

The CAPWAP plug-in CAPWAP module provide to Uni-Fy system some Radio Resource Management (RRM) functionalities. CAPWAP module is able to manage specific radio parameters (such as transmission power and channel of AP) in a dynamic, centralized and automatic way. This plug-in can assign, discover and update specific AP configuration. In order to use the plug-in in a suitable manner, APs that belong to WLAN must be able to support OpenWRT firmware.

The SIP plug-in This plug-in allows the authentication of SIP-enabled phone through the standard SIP authentication procedure, without the using of javascript-enabled browser or multi-windows capable devices (e.g. PDA). The plug-in supports two different modalities for authentication:

- **Passive** The plug-in extracts useful information from SIP messages passing through it, checking if the source/destination of the message is trusted (so letting pass the message) and if successful registrations/deregistrations occurs. In such case, it then sets properly the status of the users in the status table. It is important to note that no modification are needed on the SIP server side, letting it act with the standard behaviour.

- **Active** The plug-in checks only if the source/destination of the message is trusted (it is present in the Gatekeeper configuration file) and in this case let the message arrive to its destination. The complexity of authorizing users is instead moved into SIP servers, which requires to be modified in order to be able to create connections with the Authenticator module of Uni-Fy and send it proper messages (see section 3)

2.2.6 The Authentication module

The authentication module is built upon the regular TCP/IP stack. It listens to the TCP port 54273 on the wired interface. It is contacted by an authorized HTTP or SIP authentication server, and accepts authentication of users.

It has the ultimate responsibility of moving a client to the `AUTHORIZED` status, granting to it full access to the network, and to revoke it upon explicit logout or when the authorization period expires with no renewal.

It obeys four textual commands, logically partitioned into two groups: two commands are used by HTTP authentication servers and other two are used by SIP ones.

- This two following commands are those used by an HTTP authentication server willing to authorize a user and revoke such authorization respectively:

  `authenticate` *IP name email token*
  `revoke` *IP*

- This other two following commands are those used by a SIP authentication server willing to authorize a user and revoke such authorization respectively:

  `authenticate_sip` *IP name email token*
  `revoke_sip` *IP*

Note that the authentication commands provides additional information for logging, i.e. the name and email of the authenticated user and a token that has been during the DHCP process and sent to the client in the early authentication phase.

2.2.7 The Reporting module

It sits on TCP port 54272 on the wired interface and is contacted by PHP pages on the Uni-Fy machine itself in order to provide data about the current status (connected clients, authorized providers).

The same module is also invoked by the status table (via the status table listener interface) when a client's status changes. In this case, if needed, the Report module is responsible of sending the appropriate client authorization or revocation command to the Gateway.

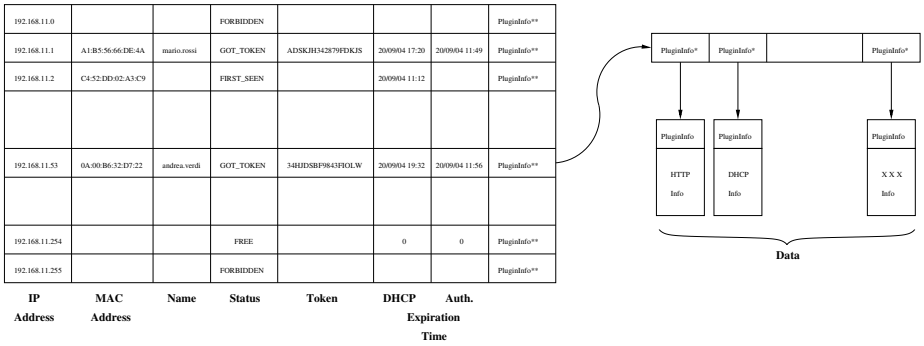


Figure 2.4: Gatekeeper status table.

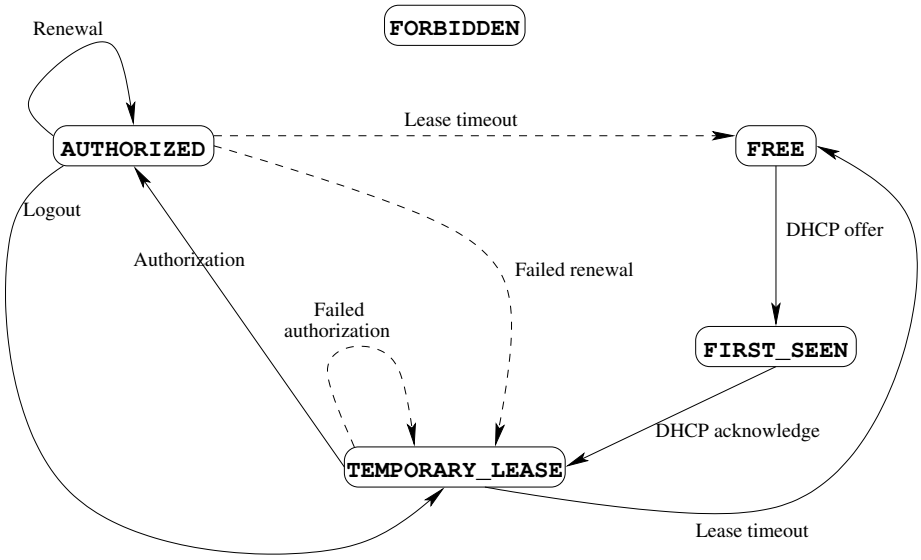


Figure 2.5: State transition diagram for an IP address association by the DHCP engine

Chapter 3

Client authentication

In a general architecture with a Uni-Fy system, a client must have a valid IP address and an authorization to use Internet services. To obtain the authorization, the client must contact his own Authentication Server and send the requested information about his own identity. If the Authentication Server recognizes the user's rights, it contacts the Uni-Fy to inform it that the client can access the Internet services.

When the client sends information to the Authentication Server it is not yet authorized by Uni-Fyfor general remote accesses to Internet, so the Uni-Fy needs proper policy to allow the forwarding of authentication-related packets.

The authentication of a user can be obtained using a web-browser based solution (e.g. captive portal) or a SIP based one. The latter is developed to support authentication od users using devices that does not support web browsing or graphical interfaces (as basic SIP wireless phone).

A somewhat simplified timing diagram for the "captive portal" authentication procedure is depicted in figure 3.1, while a simplified SIP authentication scheme is represented in figures 3.2(a) and 3.2(b).

3.1 DHCP discovery and request

When the user turns on the networking interface, the first packets that the client sends are related to dynamic configuration (DHCP) to request an IP address.

All DHCP-related packets are tunneled to the Gatekeeper, even if the requesting client is already authenticated (e.g., an IP renewal is taking place). If the Gatekeeper operates as a DHCP server, it decides the IP address to assign, whether to renew it or not, and the duration of the IP lease.

3.2 Capture method: the "captive portal" technique

When the client obtains an IP address, it still cannot browse the web. When the browser is pointed to some URL, it will first perform a DNS query, to which the Gateway is transparent (second shadowed strip of Figure 3.1). The DNS query will concern either the domain name of the requested website or, if the Proxy server has been correctly set, the proxy IP.

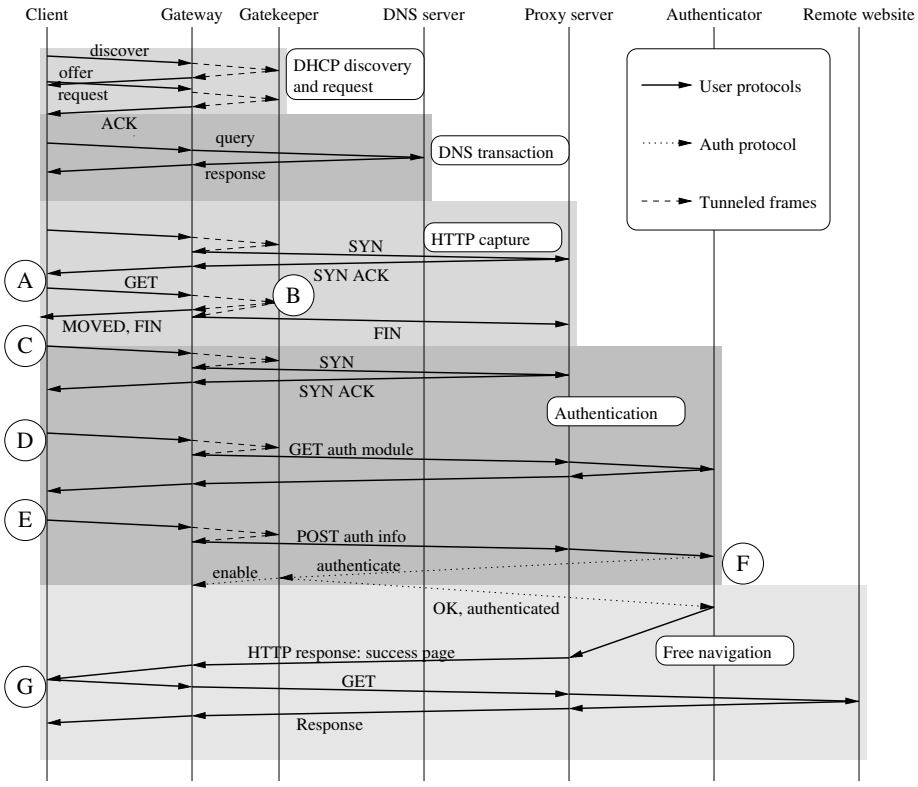


Figure 3.1: Simplified timing diagram from client appearance to full authentication

After getting the correct DNS response, the browser opens a connection to the proxy server; after the connection is open (not all the handshake is drawn in the diagram), an HTTP GET request is issued towards the proxy server (time label A). The packet containing the GET request is intercepted by the Gatekeeper at time label B, which forges a response packet by pretending to be the proxy server. The response contains an HTTP 302 code which redirects the client to an authentication page. Moreover, the packet has the FIN flag set, so that the client browser closes its connection to the proxy server. A FIN packet is also forget and is set to the proxy server on behalf of the client. As a result, at time label C the connection is closed and the client is redirected to the authentication provider's page. Closing the connection is necessary, because after forging the redirect response sequence numbers at the client and at the proxy endpoints would not correspond.

Thus, the HTTP authentication process starts with the client issuing a new web proxy connection. The next GET request (actually a CONNECT request to a secure server, time label D) is directed to the Authentication Server's login page, so it is admitted by the Gatekeeper. After getting the page, the user fills it in and (time label E) posts the login data to a PHP authentication script in the Authentication Server, which (at time label F) checks the user's login credentials (password or certificate) and, upon approval, contacts the Gatekeeper's authentication module and sends an authorization command. The Gatekeeper updates its own status table, sends an authorization command to the Gateway, unblocking the user's packets, and a response to the Authentication Server. The Authentication Server, in turn, sends a success page to the client, and the client is free to navigate the web (its connection to the net is no longer tunneled to the Gatekeeper).

3.3 Refresh of HTTP authorization

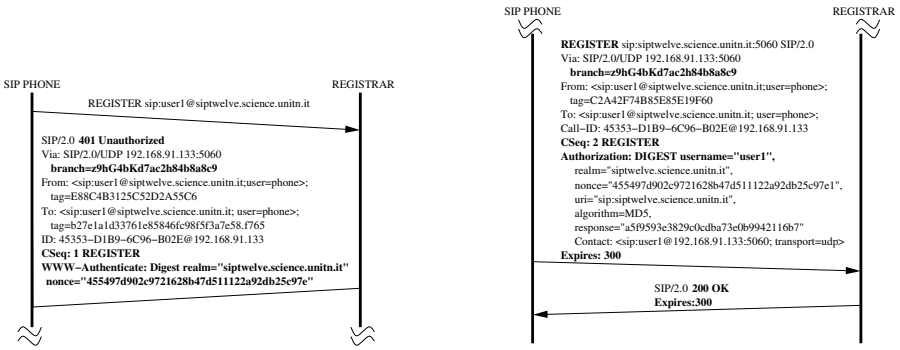
The client authorization must be periodically refreshed in order to avoid IP/MAC spoofing by unauthorized users.

The HTTP Authentication Server's "success" page contains javascript commands that open a pop-up browser window displaying a secure page that is periodically reloaded, each time passing a new shared random alphanumeric string, called "token".

The first token is created by the Gatekeeper upon DHCP request, and passed to the client when redirected to the Authentication Server. When the Authentication Server confirms the user's authorization, he sends in the user's token, which is checked against the stored value. A new value is generated and sent to the client's pop-up page, and will be requested at the first renewal. A new token is generated at every renewal.

3.4 Another authentication procedure: the SIP procedure

When the client obtains an IP address, it still cannot browse access Internet services as explained before. Till the introduction of SIP authentication procedure, clients were able to gain access to Internet only through the usual HTTP authentication, that explained in section 3.2. This procedure seems quite simple but is not so trivial when devices with Wi-Fi capabilities have limited browser-capabilities or, more worse, has no graphical display at all. This is the case of next-generation phones, equipped with 802.11 interfaces but with limited display resolution and capabilities.



(a) Transaction 1

(b) Transaction 2

Figure 3.2: Typical Authentication interactions

With the introduction of the SIP authentication procedure, a SIP-enabled phone can gain access to Internet services only switching it on, without the need of a pop-up window always open.

Once a SIP phone (or softphone) has been properly configured, first of all it associates with the configured WLAN, then it looks for a valid IP address, and then sends a REGISTER message to the SIP server. Uni-Fy knows that all connections on port 5060 are SIP session so, being the client not yet authorized, it will forward the SIP message to the Gatekeeper who will ensure that the source/destination address are trusted. If the message pass trough the Gatekeeper it then arrives to the SIP server, who will start a challenge-response with the user sending back a 401 Unauthorized. Once the response of the user is received, the SIP server checks user's credentials contained in the REGISTER message and, in case of success, responds to the user with a 200 OK. Without taking into account the intermediary role of the Uni-Fy a SIP authentication procedure could be summarized with two transactions, as outlined in figures 3.2(a) and 3.2(b).

Moreover, if the SIP server is an ACTIVE one, it must also create a tell the Authenticator to authorize the user. If instead, the SIP server is a PASSIVE one, then is the Gatekeeper who will take care of moving the user to the AUTHORIZED status.

Now, the user is able to access Internet services. It will only perform periodically a refresh of that authorization, as explained in section 3.5.

3.5 Refresh of SIP authorization

The client authorization must be periodically refreshed in order to avoid IP/MAC spoofing by unauthorized users.

The 200 OK message issued by the SIP server in response to a REGISTER one contains an expire time. It looks like a count-down timer, that once reached 0 deauthenticate the user if no renewal is performed. So, it's responsibility of the phone/softphone software to respect the SIP protocol and repeat from the beginning the authentication procedure before the renewal time expires. As usual, if the SIP server is marked ACTIVE, then it must also contact the Authenticator in order to authorize the user. Because of, the SIP authentication function on the Authenticator has been thinking to be

similar to the usual HTTP one, also in this case a new token must be generated every time an authentication/renewal is performed.

As before, the first token is created by the Gatekeeper upon DHCP request, but differently, SIP clients are not aware of the token generation. When the SIP server confirms the user's authorization, he gets the user's token from the Reporter interface and check it against the stored value in the status table. A new value is generated and stored in the status table, and will be requested at the first renewal. A new token is generated at every renewal (as usual).

Chapter 4

Radio resource management on Uni-Fy

The objective of RRM is to utilize the limited radio spectrum resources and radio network infrastructure as efficiently as possible. In order to achieve this target, RRM involves strategies and algorithms for controlling several critical parameters, such as:

- Channel allocation
- Transmit power
- Handover criteria
- Modulation scheme
- Error coding scheme

Efficient RRM schemes may increase the system capacity and efficiency more than other techniques, as example channel coding and source coding schemes. In according with type of implementation , RRM can be static or dynamic. In the first case, administrator determines, in unchanging manner, the configuration of network parameters. It's the simpler technique, but often also the less efficient. In the dynamic RRM implementation the parameters of network can be change dynamically and automatically, it's more complicated but also efficient than static implementation. Dynamic RRM schemes may adaptively adjusts the radio network parameters to the traffic load, user positions, quality of service requirements, etc. in absolutely automatic way.

4.1 Overview of CAPWAP Protocol

The CAPWAP[1] protocol foresees the interaction between two entities, Wireless Termination Points (the client that runs directly on AP) and Access Controller (the server that runs as plug-in of Uni-Fy system). The goals for the CAPWAP protocol, developed by IETF actually only as a draft, are listed below:

[1] Control And Provisioning of Wireless Access Points

- To centralize the authentication and policy enforcement functions for a wireless network. The AC may also provide centralized bridging, forwarding, and encryption of user traffic. Centralization of these functions will enable reduced cost and higher efficiency by applying the capabilities of network processing silicon to the wireless network, as in wired LANs.
- To enable shifting of the higher level protocol processing from the WTP. This leaves the time critical applications of wireless control and access in the WTP, making efficient use of the computing power available in WTPs which are the subject to severe cost pressure.
- To provide a generic encapsulation and transport mechanism, enabling the CAPWAP protocol to be applied to many access point types in the future, via a specific wireless binding.

The CAPWAP protocol is designed to be flexible, allowing it to be used for a variety of wireless technologies. The protocol provides also advanced features in order to efficiently manage wireless devices, such as:

- Configuration of radio parameters
- Collection of operation's statistics
- Devices firmware upgrade
- Manage of wireless clients

The CAPWAP plug-in supports a subset of functionalities defined in the IETF draft, in order to integrate dynamic RRM features on Uni-Fy system.

4.2 Uni-Fy CAPWAP plug-in structure

The aim of CAPWAP plug-in is to propose a basic support in order to integrate dynamic RRM features on Uni-Fy system. The CAPWAP plug-in extended the Uni-Fy's functionalities and it lets to manage and control the specific radio interfaces parameters. It is the first plug-in that allows to manage the network parameters, others only deals with client ambit. In particular WTP Table provides to others modules, such as Status Table with clients, all relevant informations about status and AP configurations that makes the Uni-Fy network. For each AP you can specifies:

- **Nickname** of AP, useful in WLAN with many APs.
- **Location** of AP, useful in order to regroup APs.
- **IP** address of AP, necessary in order to identifies the AP.
- **Radio Channel** that AP must use.
- **Transmission Power** that AP must use.

Nowadays AP parameters can be edit in a text file, saved on Gatekeepermachines, and loads automatic and dynamically on APs, though in the futures may be possible create more sophisticated programs and algorithms in order to manage radio resources in more efficiently way. The CAPWAP plug-in can provide a basic and essential support

to a new series of services, that can be perform, for examples, dynamic load balancing, auto-configuration of AP, dynamic channel allocation in order reduce co-channel interference etc. The CAPWAP plug-in opens de-facto a new sector of advanced services, that performs radio resource management, directly managed by Uni-Fy system. The structure of CAPWAP plug-in, likewise others plug-in, is designed to be modular and extendible. Like shows in figure 2.1, CAPWAP plug-in is split in two different modules, CAPWAP A and CAPWAP B. CAPWAP A manages only the packets of unassociated AP, instead CAPWAP B only the packets that comes from associated AP. This splitting allows that, in the future, association procedure are managed by external tool. WTP Table contains status and WLAN configuration. In particular, each entry contains the specific configuration of each AP, such as its status, radio channel and its transmission power. WTP Configuration module waits, on configurable UPD port, a packet that forces the re-read of WTP parameters. If WTP Configuration module receives this packet, contacts CAPWAP B that reloads the WTP configurations.

4.3 Overview of OpenWRT

OpenWrt is described as a Linux distribution for embedded devices. Instead of trying to create a single, static firmware, OpenWrt provides a fully writable filesystem with package management. This frees you from the application selection and configuration provided by the vendor and allows you to customize the device through the use of packages to suit any application. In order to use CAPWAP plug-in, AP must be updated with OpenWRT firmware and with wtp program, included in Uni-Fysources. The wtp program is able to communicate with Uni-Fy CAPWAP plug-in in order to provide RRM features. In order to run wtp program on your AP you must update it with OpenWrt firmware provided with Uni-Fy sources. There are multiple ways to reflash the firmware, we will explain each method below.

- **via vendor supplied web interface**: This is the easiest method, Open your web browser and use the firmware upgrade page on your device to upload the OpenWrt firmware.
- **via tftp**: If you're being extremely cautious or are attempting to reflash from a failed upgrade, you can use tftp to install the firmware.
- **via CFE**: you needs the serial cable.
- **via JTAG**: It's not recommended to flash the kernel image via jtag, as it will take more than 2 hours.
- **via the OpenWrt commandline**: Reflashing OpenWrt will overwrite the filesystem, erasing all previous applications and data.

You can use any method, the end result will be the same. After reflashing, the device will automatically reboot into the new firmware. If you are not happy with OpenWrt, you can always reinstall your original firmware. Please be sure you have it downloaded and saved on your PC. For more details visit the official web[2] page of OpenWRT project In order to run wtp program, you must connect with SSH session to AP, then move in the `/etc/sbin` directory and digit `./wtp CAPWAP.pmt`. The CAPWAP.pmt is a text file that contains the AP's parameters, for more detail see also 6.6.

[2] www.openwrt.org

4.4 CAPWAP plug-in procedure

The procedure of CAPWAP, as shows in figure 4.1, are described below:

1. **Discovery Phase**: The CAPWAP Protocol begins with a discovery phase. The AP sends a Discovery Request message, Uni-Fy CAPWAP A plug-in respond with a Discovery Response message. From the Discovery Response messages received, a AP will select the server with which to request service.

2. **Join Phase**: The AP sends a Join Request message in order to request service from the specific server, in this case the Uni-Fy CAPWAP B plug-in, that responds with a Join Response message.

3. **Configuration Phase**: The AP sends a Configuration Status Request message where it suggests a possible configuration. CAPWAP B plug-incan confirms the configuration responding with Configuration Status Response message or propose a new configuration with Configuration Update Request message. In the last case AP must respond with Configuration Update Response message in order to confirm the new configuration and then it can resend the Configuration Status Request message.

4. **Run Phase**: If all previous phases are ended correctly, AP enter in Run state and stars to provide service to wireless client, in this state it periodically sends to CAPWAP B plug-inEcho Request message in order to verify the state of connection. CAPWAP B plug-inresponds with a Echo Response message.

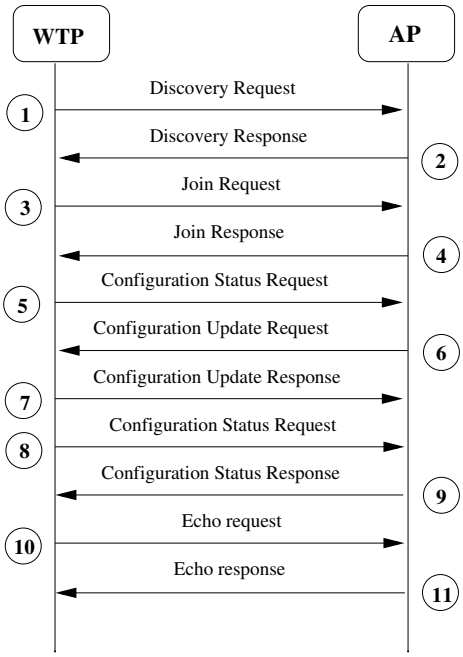


Figure 4.1: CAPWAP session

Chapter 5

Installation and configuration

5.1 Getting the source code

The source code is available from Twelve website http://twelve.dit.unitn.it/tools.html. Please contact

`brunato@dit.unitn.it`

to get all download parameters.

After performing the source checkout, all necessary code is contained in the `wilmaproject/gateway` subfolder, which shall be referred to as the "root" directory of the program. The root directory contains the following items:

- `Common/` contains source code which is used both by the Gateway and by the Gatekeeper programs.
- `Gateway/` contains the Gatewayspecific source code.
- `GateKeeper/` contains the Gatekeeperspecific source code.
- `parameters/` contains the runtime configuration files for the two programs.
- `web.TEMPLATE/` contains web page templates and code for provider choice, for authentication and for system interrogation.
- `SER/` contains Uni-Fy module for Sip Express Router (SER) and a sample configuration
- `specs/` contains system documentation and specifications (and this document itself).
- `Makefile` is the overall makefile for the whole project.
- `Doxyfile` is the Doxygen configuration file for automatic API documentation.
- Some `restart` scripts to automate the execution of the programs.

5.2 Compiling the system

The system has been tested on Linux (Redhat 9 and Debian distributions) and Cygwin machines. The following packages must be installed:

- GNU C++ compiler (others have not been tested);
- Standard libraries.
  On Linux systems:
  - glibc (no particular requirements about version, probably also libc5 is good, please send us feedback);
  - libpthread (for POSIX threads, it should be present by default);
  - libpcap version 0.7 or greater (for layer 2 packet capture, also present by default on many systems). Note that versions 0.6 and below miss some key functionalities.

  On Cygwin systems:
  - the cygwin DDLs
  - libpcap for Windows, retrievable from

    `http://winpcap.polito.it/`

    The system has been tested with version 3.1beta3. Please be sure to download both the installer and the developer's pack. After installing it, please edit `Gateway/Makefile` and set the correct path in `WPCAP_PATH` for winpcap includes and libraries.

If all requirements are satisfied, compilation is simply achieved by tiping `make` at the root directory. No errors or warnings should appear: please notify us of any compiler warning at the contact address provided above.

Compilation of the full documentation (this manual and API description) requires LaTeX and the Doxygen documentation system. The latter can be found at

`http://www.stack.nl/~dimitri/doxygen/`

RPMs and other distributions are probably included in OS installation CDs. The Cygwin version can be selected in the standard Setup program. To generate documentation, type

```
make doc
```

from the home directory. The `doc` directory shall be generated with both HTML and LaTeX contents, while the present document will be created in the `specs` directory.

5.3 Configuration

Next, the system must be configured. Please refer to Figure 5.1 for interface names:

- The "inner" interface, i.e. the Gateway interface to the wireless LAN.
- The Gateway's tunnel endpoint interface to the Gatekeeper.

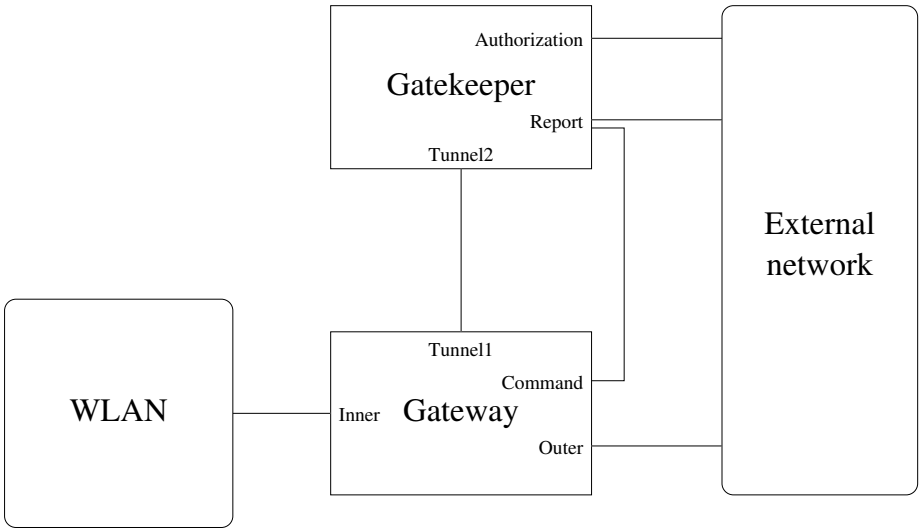


Figure 5.1: Reference for configuration

- The "outer" interface, i.e. the Gateway interface to the external network.
- The Gateway "Command" interface, contacted by the Gatekeeper in order to change a client's status.
- The Gatekeeper's tunnel endpoint interface to the Gateway.
- The Gatekeeper report interface.
- The Gatekeeper authorization interface.

In most systems, some of these interface correspond to the same physical device. Usually, interfaces "Tunnel1", "Outer" and "Command" are the same (the Gatekeeper resides on the outer network segment of the Gateway), while the Gatekeeper has only one interface, so that "Tunnel2", "Report" and "Authorization" are equal.

If the Gateway and the Gatekeeper reside on the same machine, then interfaces "Tunnel1", "Tunnel2" and "Command" are local (their IP is 127.0.0.1), while interfaces "Outer", "Report" and "Authorization" coincide.

After identifying each physical interface and associating it to the ones reported in Figure 5.1, the IP address of each interface must be found.

Only for interfaces "Inner" and "Outer", also the system name is needed. On UNIX systems, the name is `eth`n, where n is a small index; all required information can be borrowed from the output of `ifconfig`. On Cygwin, two names are required for each interface: the first is recognized by the Cygwin networking primitives, and is similar to the UNIX name; the second is the Windows resource name, beginning by "`\Device...`" used by Winpcap, and can be obtained from the capture menu of Ethereal (`www.ethereal.com`).

Next, a pool of IP addresses must be allocated for the DHCP server. The usual way to do this is ask to the network administrator of your structure. Currently, the system works for a subset of a class C network (255 addresses), some more work is required to extend it to a larger IP address pool.

5.3.1 Interface constraints

Some conditions must be met by the network interfaces of the system.

- The IP of the "Inner" interface of the Gateway must lie in the same class C network as the DHCP pool.
- The "External Network" must contain a web proxy server, a web server. If necessary, also a DNS server in the Gatekeeper machine and a SIP server are required.
- Two more addresses in the DHCP subnet must be reserved:
  - the gateway that shall be declared by the DHCP server,
  - the DHCP server itself.

  Both IPs shall be managed (actually, faked) by the Gateway program through the "Inner" interface, and must not be assigned to any network card.
- The Gatekeeper's "Tunnel2" and "Report" IPs must be accessible from the Gateway.
- The Gatekeeper's "Report" IP must be accessible from the local webserver.
- The Gatekeeper's "Authorization" interface must be accessible from the Authentication providers, i.e. from the outside (unless authentication is managed internally).
- The Gateway's "Tunnel1" interface must be reachable from the Gatekeeper.
- The Gateway's "Command" interface must be reachable from the Gatekeeper.
- All SIP authentication servers must be reachable from the Gatekeeper.
- The Gatekeeper's "Authorization" and "Report" interfaces must be accessible via a secure tunnel from ACTIVE Sip Authenticators, if any.

5.3.2 Gateway configuration

Edit a file in the `parameters` directory having a `.gw` extension (use one of the enclosed templates). The file format is free, only "equal" ("=") signs are important and introduce the values of the configuration parameters, that must be reported in the given sequence. Ports should remain as they are. Parameters have the following meaning:

Inner interface configuration

- **Inner interface network** — The class C network containing the DHCP pool.
- **Inner interface netmask** — The corresponding netmask.
- **Inner interface ARP domain** — The inner interface must respond to ARP queries for the class C network gateway, i.e. the first of the two reserved and unassigned addresses in 5.3.1.
- **Inner interface ARP mask** — Usually, the above address is the only one that must respond to ARP queries, so a full mask is adequate.

- **Inner interface properties name** — The inner device name.
- **Inner interface capture name** — The inner device name for pcap (same as above in UNIX systems, different for winpcap).

Outer interface configuration

- **Outer interface network** — The network accessible via the outer interface is the whole world (in principle), so a null address is usually good.
- **Outer interface netmask** — For the same reason, a null mask is adequate.
- **Outer interface ARP domain** — The "outer" interface must respond to ARP queries on behalf of wireless clients in order to work as gateway, so the class C subnet containing the DHCP pool should usually be written.
- **Outer interface ARP mask** — The corresponding class C netmask.
- **Outer interface properties name** — The outer device name.
- **Outer interface capture name** — The outer device name for pcap (same as above in UNIX systems, different for winpcap).
- **Outer interface gateway** — The IP address of the external network gateway.

Interfaces towards the Gatekeeper

- **Tunnel IP** — The IP address of the "Tunnel1" interface.
- **Tunnel listening UDP port** — The corresponding UDP port number.
- **Command interface listening IP** — The IP address of the "Command" interface.
- **Command interface listening TCP port** — The corresponding TCP port number.
- **Gatekeeper's tunnel IP** — The IP address of the "Tunnel2" interface, as reported in the Gatekeeper configuration (see below).
- **Gatekeeper tunnel listening port** — The corresponding TCP port number.
- **Gatekeeper's reporter IP** — The IP address of the "Report" interface, as reported in the Gatekeeper configuration (see below).
- **Gatekeeper reporter listening port** — The corresponding TCP port number.

Dropped packet sink application

- **Dropped packet sink IP** — The router can optionally send all dropped ethernet frames to an external application listening to a specified IP address and UDP port.
- **Dropped packet sink UDP port** — UDP port of the sink application.

If no sink is required, then the first parameter must be set to 0.0.0.0, the second to 0.

External network parameters

- **DNS servers** — A semicolon-separated list of IP addresses of authorized DNS servers.
- **Initial HTTP server** — IP address of the initial web server for authentication purposes.
- **Proxy servers** — A semicolon-separated list of IP addresses of the authorized Proxy servers. The only supported port at this moment is port 3128.

SIP routing policy

- **Submit all SIP packets to Gatekeeper** — Determination of the SIP rule in the ruletable (1 to forward all SIP messages to the gatekeeper, 0 to forward only SIP unauthorized user messages)

Runtime mode

- **Run mode** — Determination of the runtime mode (1 to detach from console and run as daemon, 0 otherwise)
- **Log file** — Output destination for daemon mode.
- **Transparent mode** — Should be kept to zero. A nonzero value lets all packets flow through the gateway without authentication (only DHCP requests are treated), and can be used for debugging and testing.
- **PID file** — File where the process ID must be written to (may be left blank).

5.3.3 Gatekeeper configuration

Edit a file in the `parameters` directory having a `.gk` extension (use one of the enclosed templates). The file format is free, only "equal" ("=") signs are important and introduce the values of the configuration parameters, that must be reported in the given sequence. Ports should remain as they are. Parameters have the following meaning:

Gatekeeper IP address configuration

- **Authorization IP** — The IP address of the "Authorization" interface. Note that the authorization IP address must be reachable from the authentication providers, so in most cases it should be exposed to the Internet on a public IP address (this is not the case of a self-contained system).
- **Authorization port** — The TCP port.
- **Report IP** — The IP address of the "Report" interface. The interface is also used by the `choose.php` web page to obtain the list of available authentication providers. Therefore, the IP address should be reachable from the local web server.
- **Report port** — The TCP port.
- **Dispatcher IP** — The IP address of the "Tunnel2" interface.
- **Dispatcher listening port** — The UDP port.

Gateway IP address configuration

- **Gateway tunnel IP** — The IP address of the "Tunnel1" interface of the Gateway.
- **Gateway tunnel port** — The UDP port.
- **Gateway command IP** — The IP address of the "Command" interface of the Gateway.
- **Gateway command port** — The TCP port.

DHCP information

- **DHCP class C network IP** — The base network address of the DHCP pool.
- **DHCP first assignable IP** — The smallest IP address that can be assigned by the DHCP plug-in; remember that some addresses are reserved.
- **DHCP last assignable IP** — The largest IP address in the DHCP pool.
- **IP of DNS server** — The IP address of the DNS server.
- **IP of wireless gateway** — The reserved IP used as wireless gateway.
- **IP network mask** — The class C network mask 255.255.255.0.
- **IP network domain** — Domain name of the network.
- **Fake address of the DHCP server** — The reserved IP used as DHCP server.
- **Short lease time** — The lease time (in seconds) for unauthorized clients. Usually in the tens.
- **Long lease time** — The lease time (in seconds) for authorized clients. Usually in the thousands.

Capture information

The following info applies to the captive portal method, currently the only available in the system.

- **redirection URL** — URL towards which an unauthorized client must be directed. See 5.3.4 for details.
- **redirection URL when no proxy is set** — If the client's proxy is not set, then a warning webpage should be displayed. Its URL must be reported here.
- **Proxy configuration file URL** — If the (Microsoft proprietary?) proxy configuration feature of the proxy server is used, this field must contain the URL of the `WPAD.DAT` configuration file, if possible in the gateway's local HTTP server (usually in the `local` branch, where a sample file can be found).
- **Authorization renewal time** — Expiration time (in seconds) of an authorization. A client must renew his authorization before it expires, otherwise the whole authentication process will begin.

Files

- **Run mode** — Type 1 to detach the process from the console (run in daemon mode), 0 otherwise.
- **Status table dump file** — File containing an up-to-date version of the status table, used to recover the relevant system status information after a crash or an interruption.
- **Program log file** — File containing the log of the system.
- **PID file** — File where the process ID must be written to (may be left blank).

DHCP Relay Information

- **RelayActivated** — Type 1 to use DHCP Realay agent to interact with external DHCP server. Type 0 if Uni-Fy built-in DHCP server is used (if 0, following parameters are not required).
- **DHCPServerIP** — IP address of DHCP server in external LAN.
- **OuterInterfaceIP** — IP address of interface that interacts with DHCP server (Authorization or Report IP address).
- **NatPublicIP** — IP address of interface that interacts with DHCP server. It must be equal to OuterInterfaceIP when DHCP does not pass through NAT.
- **InnerInterfaceIP** — IP address of interface that interacts with internal LAN. The gateway for internal network.
- **DHCPServerListenerPort** — Port number of the listener. Default is 67 (defined by standard)
- **DHCPClientThinkServerPort** — Port number for information that must be sent to client. Default is 67 (defined by standard)
- **InnerInterfaceName** — The inner device name.

Authentication providers

This section of the file contains one quadruple for each authentication provider.

- **Name** — Name of the provider, to be displayed in the choice page.
- **URL** — URL of the provider's authentication page.
- **Prefix** — Prefix of all URLs of that provider that must be accessed by unauthorized clients.
- **Domain** — The canonical domain name of the host, usually the domain part of the above URL.
- **IP** — IP of the provider's web server.

A provider record with an empty URL field has a special meaning: it is the description of the access provider choice page.

After the last provider, an empty "Name" field ends the list.

SIP Plugin Parameters

The following parameters tells Uni-Fy if enable or not the SIP based authentication feature and, in the former case, when to use a verbose SIP logging or not.

- **Enable SIP Plugin** — Type 1 to enable SIP authentication capabilities. Type 0 to not load SIP plug-in into dispatcher, disabling all SIP messages processing.
- **Debugging mode** — Type 1 to enable a verbose logging of the SIP message processing. Type 0 to have only a minimal SIP processing logging.

SIP Servers

The section contains one triplet of data for each SIP authentication server. The plug-in handles only UDP traffic, thus a SIP phone using TCP will be unable to work.

- **Name** — The name to assign to a specific SIP server
- **IP address** — The IP address of the SIP server
- **Authentication mode** — The working mode of the SIP server. 0 specifies that the SIP server works in PASSIVE mode. 1 specifies that the SIP server works in ACTIVE mode

As for Authentication provider list, an empty "Name" field ends the list.

Moreover, you should be very careful configuring SIP servers. A SIP server marked as PASSIVE but acting as ACTIVE can bring the system to unpredictable results, and a SIP server marked as ACTIVE but acting as PASSIVE will never be able to authorize users.

CAPWAP parameters

This section of the file contains the four parameters needed by the CAPWAP plug-in.

- **Enable CAPWAP Plugin** — Type 1 to use CAPWAP module in order to manage Radio interfaces of APs. Type 0 otherwise.
- **Path of parameters file** — Path of file that contains AP configuration, an example of this file can be found in /parameters/TEMPLATE.cw.
- **IP address of configuration service** — IP address of configuration service. This service listens to the interface (and a dedicated port) for receiving of signals that force update of AP configurations.
- **UDP port of configuration service** — Number of UDP port of configuration service.

Perpetually authorized clients

Following, a section with perpetually authorized, therefore this section must be handled with much care.

Note: from Version UniFy-20060412 there is a separation between authorization and IP assignment. A user that must insert user-id and password to provide his own credentials, receives always a dynamic IP address (both built-in DHCP server and DHCP Relay is used) and it is not in the following list. A perpetually authorized user can receive a dynamic IP address or can have a static IP address on his own machine.

- **Name** — Name of the client.
- **Email** — Email address of the person responsible for the client.
- **MAC** — Ethernet MAC address of the client.
- **IP — built-in DHCP server**

  - IP address that must be attributed to the client, if the client require static configuration or reserved IP.
  - 0.0.0.0 if the IP address is requested trough DHCP transaction.

  DHCP Relay server

  - IP address that must be attributed to the client, if the client has manual configuration of IP in the machine.
  - 0.0.0.0 if the IP address is requested through DHCP transaction. Related to configuration in DHCP server, the provided IP address can be always the same or can change.

After the last static IP, an empty "Name" field ends the list.

5.3.4 Web server configuration

The "captive portal" authentication method requires the presence of one or more web servers, that must be accessible via the proxy server.

The web server must be configured so that an alias is hooked to the `web` subdirectory of the system. Three directories are contained in `web`:

- `local` contains all pages related to the first access of an unauthorized user: instructions about setting the proxy, choice of the authentication provider.
- `auth` contains all pages related to authentication and authorization renewal.
- `status` contains all pages that can be used to show the system status.

Configuring the access server

The access server can be located in the Gateway or in the Gatekeeper machine. It must expose the `local` directory. The only configuration needed is in file `choose.php`, where the correct port number must be set in variable `gk_reporter_port`.

If the automatic proxy configuration feature is activated, it is important to expose the `local` directory via the wireless interface of the gateway. In this case, the `wpad.dat` file must be configured with the correct proxy parameters.

Optionally, the `status` directory can be exposed. In this case, files `index.php` and `client.php` must connect to the "Report" interface of the Gatekeeper.

Configuring an authentication provider

The authentication provider must expose the subdirectory `web/auth` via a secure connection (https).

An authentication provider must contain some user authentication data. These can be provided by a RADIUS or LDAP service, by a MySQL database or in an internal table, or by any combination of such services. For instance, a centralized LDAP service can provide authentication for all regular users, while some temporary users can be managed by a MySQL table in a local machine.

Description of available services is provided in the `config/config.inc.php`, that must be configured according to the chosen authentication system.

Two more pages are provided in `config` to add people to a MySQL database and to change password.

If communication ports are changed with respect to their default values (ranging from 54270 to 54273), then all PHP pages must be modified accordingly.

5.3.5 SIP server configuration

Enabling SIP based authentication method requires the presence of one or more SIP registrar/location servers, that must be accessible from the Gatekeeper.

The Uni-Fy system is able to support two authentication procedures:

- **Passive** If the SIP server is marked as PASSIVE in the Gatekeeper configuration file, then it must only check user credentials contained in REGISTER messages and respond the user consequently
- **Active** If the SIP server is marked as ACTIVE in the Gatekeeper configuration file, then it must check user credentials as usual but, in case of success, it must also retrieve information about that user contacting the Reporter module of Uni-Fy, contact the Authenticator module and authorize him. Thus, an ACTIVE SIP server must be customized in order to be able to contact Uni-Fy if successful registration/deregistration occurs.

  From a security point of view, it is important that connections between SIP servers and Reporter/Authenticator modules of Uni-Fy are encrypted (e.g. using secure tunnels).

In case of incorrect user credentials, either ACTIVE either PASSIVE SIP servers must act in the usual manner, responding the user with the adeguate message error.

Any SIP authentication server must interface itself with some authentication data, such as a RADIUS or LDAP service, a MySQL database, internal tables or any.

Moreover, it is also strongly recommended paying attention when marking SIP servers as Active or Passive: a server working in ACTIVE mode (in the sense explained above) but marked as PASSIVE will lead to unpredictable results.

An example of working configurations (both PASSIVE and ACTIVE) for SER SIP server is present in the `SER` directory. More information about how to configure the server as PASSIVE or ACTIVE can be found in the `SER\README` file.

5.3.6 Configuring AP parameters

If CAPWAP plug-inis enabled, you must compose the AP configuration file. The structure template of this file, Template.cw, is contained in parameters directory.

- **AC Name** — A nickname of AC, useful in WLAN with multiple AC.
- **ESSID** — The ESSID that AP must use.
- **Netmask** — The netmask that AP must use, it can be 0.0.0.0.
- **Gateway** — IP address of AP's gateway, it can be 0.0.0.0.
- **Out UDP Port** — UDP Port for outcoming packets, dafault value 54280.
- **In UDP Port** — UDP Port for incoming packets, dafault value 54281.
  The follows parameters must be repeats for each AP.
- **Name** — A nickname of AP.
- **Location** — Location of AP.
- **IP** — IP address of AP.
- **Channel** — Radio channel that AP must use.
- **Transmission Power** — Transmission power that AP must use.

5.4 Running the programs

From the base directory, the programs can be launched with the following commands:

```
GateKeeper/gatekeeper parameters/[gatekeeper config].gk
Gateway/gateway parameters/[gateway config].gw
```

It is important to start the Gatekeeper program *before* the Gateway. Both programs output a log, either to the standard output or to the specified log file, according to the specified run mode. Events such as DHCP offers and acknowledgments, authorizations and revocations are recorded in the log by the Gatekeeper. When authorization to a client is revoked, the Gatekeeper logs the traffic generated by that client during that session.

To ease program restart, the bash shell script "`restart.sh`" is provided in the main directory to be configured (`CONFIGBASENAME` variable). When launched, it will stop any gateway or gatekeeper program instance and launch the programs in the correct order. If the previous program termination left any dangling TCP socket, the script will repeatedly attempt to restart the program until socket timeouts free the required resources.

In order to ensure higher availability, the shell script "`check_if_running`" is provided; modify its parameters according to the installation and make `cron` invoke it periodically by adding a line to the root's crontab (command "`crontab -e`") such as

```
0,5,10,15,20,25,30,35,40,45,50,55 * * * * /root/bin/check_if_running
```

Chapter 6

Examples of Network Configuration

The Uni-Fy system can be configured in different topologies: with the Gatekeeper and Gateway programs installed on a single machine or on two machines.

In this section, *public IP address* means an IP address that can be managed by the external LAN; in particular, if the external network is the Internet, the term has its common meaning, and the *public IP address* must be a real public IP address. On the other hand, if the external LAN is a private LAN with a border gateway and a proxy server, the *public IP address* can be any IP address that can be routed in the LAN.

In this section we focus on five configurations:

I The Gateway and the Gatekeeper are in the same server (see Figure 6.1), with CAPWAP and SIP plug-ins disabled
Required software

- Uni-Fy software
- web-server (e.g. Apache2)

Required IP addresses

- one public IP address
- a private class of IP addresses that can be routed in the LAN

II The Gateway and the Gatekeeper are in two different server and both machines have a public IP address (see Figure 6.2). CAPWAP and SIP plug-ins are disabled.
Required software

- Gateway software (in Server1)
- web-server (e.g. Apache2) (in Server1)
- Gatekeeper software (in Server2)
- proxy software (e.g. Squid) (in Server3)

Required IP address

- two public IP address

- a private class of IP addresses that can be managed by the network (the traffic must be routed in the LAN)

III The Gateway and the Gatekeeper are in two different servers and only one machine has a public IP address (see Figure 6.3). CAPWAP and SIP plug-ins are disabled.
Required software

- Gateway software (in Server1)
- web-server (e.g. Apache2) (in Server1)
- Gatekeeper software (in Server2)
- proxy software (e.g. Squid) (in Server2)

Required IP address

- one public IP address
- a private class of IP address without requirements: the traffic from the LAN is managed by the proxy so the class needs not be routable in the LAN.

IV The Gateway and the Gatekeeper are in the same server (see Figure 6.1) (as Configuration **I**) with DHCP Relay Agent. CAPWAP and SIP plug-ins are instead disabled.
Required software

- Uni-Fy software
- web-server (e.g. Apache2)

Required IP addresses

- one public IP address
- a private class of IP addresses that can be routed in the LAN

V The Gateway and the Gatekeeper are in the same server (as Configuration **I**), but CAPWAP and SIP plug-ins are enabled (see Figure 6.4). It is important to note that Uni-Fy system with enabled SIP based authentication could, in priciple, avoid the captive portal technique. So, in the future, Uni-Fy could also give the possibility of optionally disabling HTTP based authentication.
Required software

- Uni-Fy software
- web-server (e.g. Apache2)
- at least one AP equipped with OpenWRT firmware (for a useful usage of CAPWAP plug-in)

Required IP addresses

- one public IP address
- a private class of IP addresses that can be routed in the LAN

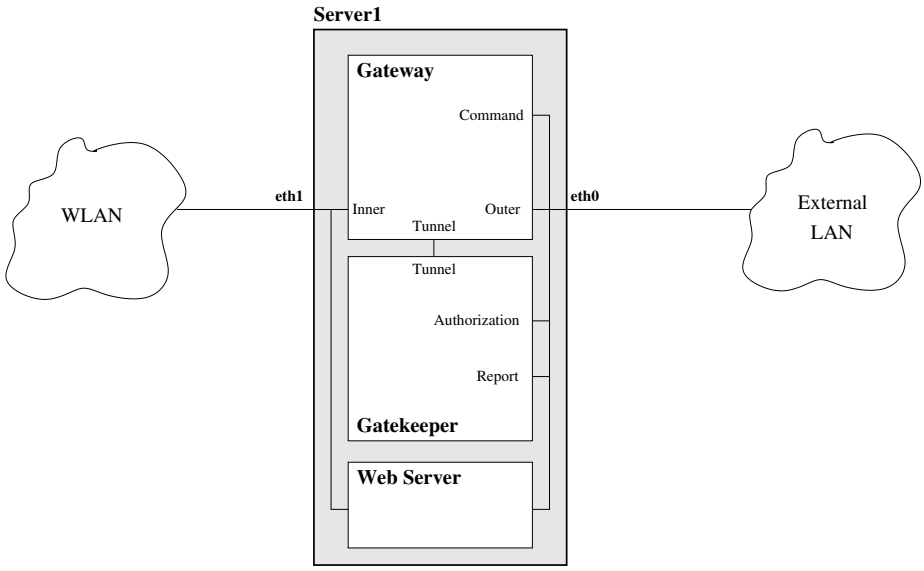


Figure 6.1: Network configuration for example 1.

6.1 Type I configuration

In this case the *Server1* contains all the required software: Gateway, Gatekeeper, and webserver.

The proxy server is required if the IP address used for wireless network cannot be managed by the external LAN.

The IP address in this example are:

- **192.168.10.2** the IP address of **eth0** interface of **Server 1**
- **172.31.194.3** the IP address of **eth1** interface of **Server 1**
- **172.31.194.0** the IP address class for wireless LAN

This configuration is an operative implementation of the authentication system. The IP address of eth0 interface is not public: it is a private static IP address because the system is running in a private network that is able to route it. Besides, the class of IP address uses for the WLAN is managed by the External LAN, so in this configuration the involved proxy server is the proxy server in the External LAN.

6.1.1 How to configure the system

Gateway

The configuration file for Gateway can be found in the *Example* directory: "configuration1.gw". This file must be saved in the *parameter* to force the system to read it during the start-up.

```
configuration1.gw
Gateway configuration for the actual faculty server.
```

```
Inner interface network = 172.31.194.0
Inner interface netmask = 255.255.255.0
Inner interface ARP domain = 172.31.194.1
Inner interface ARP mask = 255.255.255.255
Inner interface properties name = eth1
Inner interface capture name = eth1

Outer interface network = 0.0.0.0
Outer interface netmask = 0.0.0.0
Outer interface ARP domain = 172.31.194.0
Outer interface ARP mask = 255.255.255.0
Outer interface properties name = eth0
Outer interface capture name = eth0
Outer interface gateway = 192.168.10.1

Tunnel IP (for gatekeeper communications) = 127.0.0.1
Tunnel listening UDP port = 54270
Command interface listening IP = 127.0.0.1
Command interface listening TCP port = 54270
Gatekeeper's tunnel IP = 127.0.0.1
Gatekeeper tunnel listening port = 54271
Gatekeeper's reporter IP = 192.168.10.2
Gatekeeper reporter listening port = 54272

Dropped packet sink IP = 0.0.0.0
Dropped packet sink UDP port (0 if none) = 0

DNS servers (semicolon-separated list)
     = 193.205.194.23;192.168.121.2;192.168.121.3
Initial HTTP server = 172.31.194.3
Proxy servers (semicolon-separated list)
     = 193.205.213.166;193.205.206.25
RADIUS server = 192.168.194.168

Submit all SIP packets to Gatekeeper?
(0: No, 1: Yes) = 0

Run mode (0: console; 1: daemon) = 1
Log file (only for daemon) = gateway.log
Transparent mode = 0
PID file = /var/run/gateway.pid
```

Gatekeeper

The configuration file for Gatekeeper can be found in the *Example* directory: "configuration1.gk". This file must be saved in the *parameter* to force the system to read it during the start-up.

```
configuration1.gk
Gatekeeper configuration for the actual faculty server.
```

```
Gatekeeper IP address configuration

  Authorization IP = 192.168.10.2
  Authorization port = 54273
  Authorization relay IP
  (same as above if relay not present) = 193.205.213.112

  Report IP = 192.168.10.2
  Report port = 54272

  Dispatcher IP
  (gatekeeper's tunnel endpoint) = 127.0.0.1
  Dispatcher listening port = 54271

Gateway IP address configuration

  Gateway tunnel IP for gatekeeper communications
    (gateway's tunnel endpoint) = 127.0.0.1
  Gateway tunnel UDP port = 54270
  Gateway command interfaceIP = 127.0.0.1
  Gateway command TCP port = 54270

DHCP information

  DHCP class C network IP = 172.31.194.0
  DHCP first assignable IP = 172.31.194.51
  DHCP last assignable IP = 172.31.194.253
  IP of DNS server = 193.205.194.23
  IP of gateway (seen by wireless hosts) = 172.31.194.1
  IP network mask = 255.255.255.0
  IP network domain = science.unitn.it
  Fake address of the DHCP server = 192.168.10.2
  Short lease time (for unauthorized clients) = 180
  Long lease time (for authorized clients) = 1800

Capture information

  redirection URL
    = http://192.168.10.2/gate/local/choose.php
  redirection URL when no proxy is set
    = http://172.31.194.3/gate/local/noproxy.php
  Proxy configuration URL
    = http://172.31.194.3/gate/local/wpad.dat
  Authorization renewal time = 600

Files
  Run mode (0: console; 1: daemon) = 1
  Status table dump file = gatekeeper.dat
  Program log file = gatekeeper.log
```

```
  PID file = /var/run/gatekeeper.pid

DHCP Relay Information

  RelayActivated = 0

  DHCPServerIP =

  OuterInterfaceIP =

  (Must be equal to OuterInterfaceIP when DHCP doesn't pass throught NAT)
  NatPublicIP =

  (IP Address of the interface oriented towards WLAN)
  InnerInterfaceIP =

  (Default value is 67)
  DHCPServerListenPort =

  (Default value is 67)
  DHCPClientThinkServerPort =

  (Name of the interface oriented towards WLAN)
  InnerInterfaceName =

Authentication providers

  Name = Pimpa (EXPERIMENTAL)
  URL
    = https://pimpa.science.unitn.it/gate/auth/login.php
  Prefix = https://pimpa.science.unitn.it/gate/auth/
  Domain = pimpa.science.unitn.it
  IP = 193.205.194.149

  Name = Access
  URL =
  Prefix = http://192.168.10.2/gate/local/
  Domain = 192.168.10.2
  IP = 192.168.10.2

  Name =

SIP Plugin Parameters

  Enable SIP Plugin (0: No, 1: Yes) = 0
  Debugging mode (0: No, 1: Yes) = 0

SIP Servers

  Name =
```

```
CAPWAP Parameters:
  Enable CAPWAP Plugin (0: No, 1: Yes) = 0
  Path of parameters file =
  IP address of configuration service =
  UDP port of configuration service =

Perpetually Authorized clients

  Name = Forever_Authorized_User
  Email = Forever_Authorized_User@mail_domain
  MAC = 01:12:23:34:5A:BC
  IP = 192.168.12.10

  Name =
```

Notes

In this configuration the class of IP addresses must be routed by the external network. If you don't have a IP class with this requirement, you need a second server between the *Server1* and the *External LAN*. The connection between the two server is a point-to-point connection (with IP address without routed requirement) and the private static IP address (that can be routed by the external network) is the IP address of the *Server2* interface towards the external network. This IP address con be a public one.

For a right routing of the packets, you must add some information to route the packet:

- add a rule about the default gateway for the external interface (eth0)
  `route add default gw 192.168.10.1`

For further information about parameters of the configuration, and how to configure proxy server, web server, please refer to 6.6.

6.2 Type II configuration

In this case the *Server1* contains the Gateway software and a webserver software, while the *Server2* contains the Gatekeeper software. The *Server3* is a server proxy (e.g. Squid) and is used to show the configuration for a network without proxy; if your network has a proxy server, you can use it and this configuration requires only two machines.

The IP address in this example are:

- **192.168.0.2** the IP address of **eth0** interface of **Server 1**
- **192.168.91.137** the IP address of **eth1** interface of **Server 1**
- **192.168.11.50** the IP address of **eth2** interface of **Server 1**
- **192.168.194.166** the IP address of **eth0** interface of **Server 2**
- **192.168.11.51** the IP address of **eth1** interface of **Server 2**

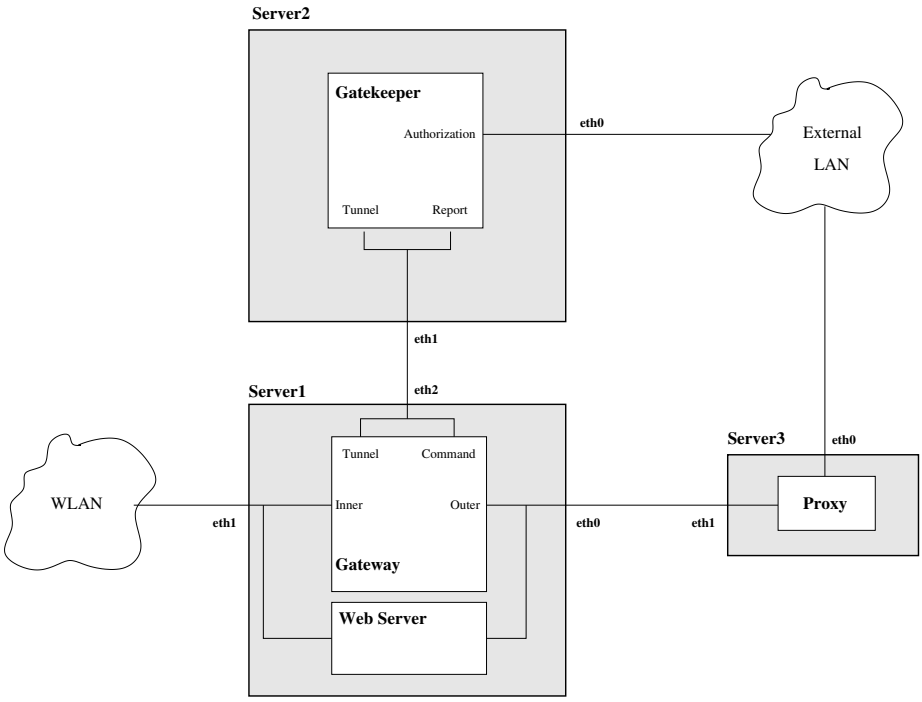


Figure 6.2: Network configuration for example 2.

- **192.168.0.3** the IP address of **eth0** interface of **Server 3**
- **193.205.194.74** the IP address of **eth1** interface of **Server 3**
- **192.168.91.0** the IP address class for wireless LAN

In this configuration the IP addresses "192.168.194.166" and "193.205.194.74" are a private static IP address and a public IP address, respectively. Both of them can be managed by the external LAN.

6.2.1 How to configure the system

Gateway

The configuration file for Gateway can be found in the *Example* directory: "configuration2.gw". This file must be saved in the *parameter* to force the system to read it during the start-up.

```
configuration2.gw
Gateway configuration for the actual faculty server.

Inner interface network = 192.168.91.136
Inner interface netmask = 255.255.255.248
Inner interface ARP domain = 192.168.91.137
Inner interface ARP mask = 255.255.255.255
Inner interface properties name = eth1
```

```
Inner interface capture name = eth1

Outer interface network = 0.0.0.0
Outer interface netmask = 0.0.0.0
Outer interface ARP domain = 192.168.91.136
Outer interface ARP mask = 255.255.255.248
Outer interface properties name = eth0
Outer interface capture name = eth0
Outer interface gateway = 192.168.0.3

Tunnel IP (for gatekeeper communications) = 192.168.11.50
Tunnel listening UDP port = 54270
Command interface listening IP = 192.168.11.50
Command interface listening TCP port = 54270
Gatekeeper's tunnel IP = 192.168.11.51
Gatekeeper tunnel listening port = 54271
Gatekeeper's reporter IP = 192.168.11.51
Gatekeeper reporter listening port = 54272

Dropped packet sink IP = 0.0.0.0
Dropped packet sink UDP port (0 if none) = 0

DNS servers (semicolon-separated list) = 193.205.194.23;
  192.168.121.2;192.168.121.3
Initial HTTP server = 192.168.91.138
Proxy servers (semicolon-separated list) = 192.168.0.3
RADIUS server = 192.168.0.3

Submit all SIP packets to Gatekeeper?
(0: No, 1: Yes) = 0

Run mode (0: console; 1: daemon) = 1
Log file (only for daemon) = gateway.log
Transparent mode = 0
PID file = /var/run/gateway.pid
```

Gatekeeper

The configuration file for Gatekeeper can be found in the *Example* directory: "configuration2.gk". This file must be saved in the *parameter* to force the system to read it during the start-up.

```
configuration2.gk
Gatekeeper IP address configuration

  Authorization IP = 192.168.194.166
  Authorization port = 54273
  Authorization realy IP
  (same as above if relay not present) = 193.205.213.112
```

```
  Report IP = 192.168.11.51
  Report port = 54272

  Dispatcher IP
  (gatekeeper's tunnel endpoint) = 192.168.11.51
  Dispatcher listening port = 54271

Gateway IP address configuration

  Gateway tunnel IP for gatekeeper communications
    (gateway's tunnel endpoint) = 192.168.11.50
  Gateway tunnel UDP port = 54270
  Gateway command interfaceIP = 192.168.11.50
  Gateway command TCP port = 54270

DHCP information

  DHCP class C network IP = 192.168.91.0
  DHCP first assignable IP = 192.168.91.141
  DHCP last assignable IP = 192.168.91.142
  IP of DNS server = 192.168.91.137
  IP of gateway (seen by wireless hosts) = 192.168.91.137
  IP network mask = 255.255.255.248
  IP network domain = mytest.domain
  Fake address of the DHCP server = 192.168.0.2
  Short lease time (for unauthorized clients) = 180
  Long lease time (for authorized clients) = 1800

Capture information

  redirection URL
    = http://192.168.0.2/gate/local/choose.php
  redirection URL when no proxy is set
    = http://192.168.91.138/gate/local/noproxy.php
  Proxy configuration URL
    = http://192.168.91.138/gate/local/wpad.dat
  Authorization renewal time = 600

Files
  Run mode (0: console; 1: daemon) = 1
  Status table dump file = gatekeeper.dat
  Program log file = gatekeeper.log
  PID file = /var/run/gatekeeper.pid

DHCP Relay Information

  RelayActivated = 0

  DHCPServerIP =
```

```
  OuterInterfaceIP =

  (Must be equal to OuterInterfaceIP when DHCP doesn't pass throught NAT)
  NatPublicIP =

  (IP Address of the interface oriented towards WLAN)
  InnerInterfaceIP =

  (Default value is 67)
  DHCPServerListenPort =

  (Default value is 67)
  DHCPClientThinkServerPort =

  (Name of the interface oriented towards WLAN)
  InnerInterfaceName =

Authentication providers

  Name = RTM (CHOOSE THIS!)
  URL = https://rtm.science.unitn.it/gate/auth/login.php
  Prefix = https://rtm.science.unitn.it/gate/auth/
  Domain = rtm.science.unitn.it
  IP = 193.205.194.21

  Name = Access
  URL =
  Prefix = http://192.168.0.2/gate/local/
  Domain = 192.168.0.2
  IP = 192.168.0.2

  Name =

SIP Plugin Parameters

  Enable SIP Plugin (0: No, 1: Yes) = 0
  Debugging mode (0: No, 1: Yes) = 0

SIP Servers

  Name =

CAPWAP Parameters:
  Enable CAPWAP Plugin (0: No, 1: Yes) = 0
  Path of parameters file =
  IP address of configuration service =
  UDP port of configuration service =


Perpetually Authorized clients
```

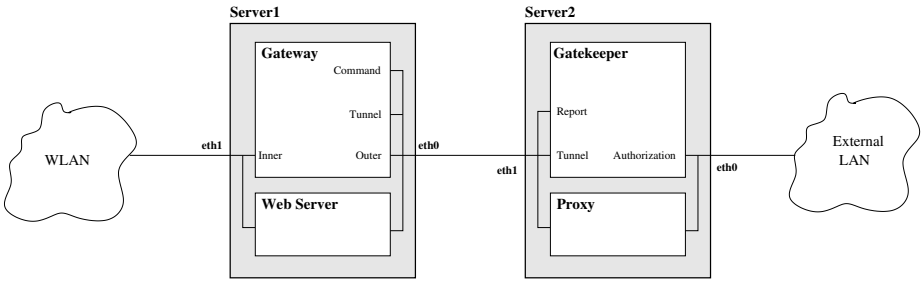


Figure 6.3: Network configuration for example 3.

```
Name = Forever_Authorized_User
Email = Forever_Authorized_User@mail_domain
MAC = 01:12:23:34:5A:BC
IP = 192.168.12.10

Name =
```

Notes

For a right routing of the packets, you must add some information to route the packet:

- add a rule about the default gateway for eth0 interface of the *Server2*
  `route add default gw 192.168.194.1 eth0`
- add a rule about the default gateway for eth0 interface of the *Server3*
  `route add default gw 193.205.194.1 eth0`
- add a rule about the private IP address for eth1 interface of the *Server3*
  `route add -net 192.168.12.0 netmask 255.255.255.0`

For further information about parameters of the configuration, and how to configure proxy server, web server, please refer to 6.6.

6.3 Type III configuration

In this case the *Server1* contains the Gateway software and a webserver software, while the *Server2* contains the Gatekeeper software and a proxy server (e.g. Squid).

The proxy server is required if the IP address used for wireless network cannot be managed by the external LAN.

The IP address in this example are:

- **192.168.12.3** the IP address of **eth0** interface of **Server 1**
- **192.168.0.2** the IP address of **eth1** interface of **Server 1**
- **192.168.0.2** the IP address of **eth0** interface of **Server 2**
- **193.205.194.74** the IP address of **eth1** interface of **Server 2** (public IP address)
- **192.168.12.0** the IP address class for wireless LAN

6.3.1 How to configure the system

Gateway

The configuration file for Gateway can be found in the *Example* directory: "configuration3.gw". This file must be saved in the *parameter* to force the system to read it during the start-up.

```
configuration3.gw
Gateway configuration for the actual faculty server.

Inner interface network = 192.168.12.0
Inner interface netmask = 255.255.255.0
Inner interface ARP domain = 192.168.12.1
Inner interface ARP mask = 255.255.255.255
Inner interface properties name = eth1
Inner interface capture name = eth1

Outer interface network = 0.0.0.0
Outer interface netmask = 0.0.0.0
Outer interface ARP domain = 192.168.12.0
Outer interface ARP mask = 255.255.255.0
Outer interface properties name = eth0
Outer interface capture name = eth0
Outer interface gateway = 192.168.0.3

Tunnel IP (for gatekeeper communications) = 192.168.0.2
Tunnel listening UDP port = 54270
Command interface listening IP = 192.168.0.2
Command interface listening TCP port = 54270
Gatekeeper's tunnel IP = 192.168.0.3
Gatekeeper tunnel listening port = 54271
Gatekeeper's reporter IP = 192.168.0.3
Gatekeeper reporter listening port = 54272

Dropped packet sink IP = 0.0.0.0
Dropped packet sink UDP port (0 if none) = 0

DNS servers (semicolon-separated list) = 192.168.12.3
Initial HTTP server = 192.168.12.3
Proxy servers (semicolon-separated list) = 192.168.0.3
RADIUS server = 192.168.0.3

Submit all SIP packets to Gatekeeper?
(0: No, 1: Yes) = 0

Run mode (0: console; 1: daemon) = 1
Log file (only for daemon) = gateway.log
Transparent mode = 0
PID file = /var/run/gateway.pid
```

Gatekeeper

The configuration file for Gatekeeper can be found in the *Example* directory: "configuration3.gk". This file must be saved in the *parameter* to force the system to read it during the start-up.

```
configuration3.gk
Gatekeeper configuration for the actual faculty server.

Gatekeeper IP address configuration

  Authorization IP = 193.205.194.74
  Authorization port = 54273
  Authorization relay IP
  (same as above if relay not present) = 193.205.213.112

  Report IP = 192.168.0.3
  Report port = 54272

  Dispatcher IP
  (gatekeeper's tunnel endpoint) = 192.168.0.3
  Dispatcher listening port = 54271

Gateway IP address configuration

  Gateway tunnel IP for gatekeeper communications
  (gateway's tunnel endpoint) = 192.168.0.2
  Gateway tunnel UDP port = 54270
  Gateway command interfaceIP = 192.168.0.2
  Gateway command TCP port = 54270

DHCP information

  DHCP class C network IP = 192.168.12.0
  DHCP first assignable IP = 192.168.12.51
  DHCP last assignable IP = 192.168.12.254
  IP of DNS server = 192.168.0.3
  IP of gateway (seen by wireless hosts) = 192.168.12.1
  IP network mask = 255.255.255.0
  IP network domain = mytest.domain
  Fake address of the DHCP server = 192.168.12.1
  Short lease time (for unauthorized clients) = 180
  Long lease time (for authorized clients) = 1800

Capture information

  redirection URL
    = http://192.168.0.2/gate/local/choose.php
  redirection URL when no proxy is set
    = http://192.168.12.3/gate/local/noproxy.php
  Proxy configuration URL
```

```
    = http://192.168.12.3/gate/local/wpad.dat
  Authorization renewal time = 600

Files
  Run mode (0: console; 1: daemon) = 1
  Status table dump file = gatekeeper.dat
  Program log file = gatekeeper.log
  PID file = /var/run/gatekeeper.pid

DHCP Relay Information

  RelayActivated = 0

  DHCPServerIP =

  OuterInterfaceIP =

  (Must be equal to OuterInterfaceIP when DHCP doesn't pass throught NAT)
  NatPublicIP =

  (IP Address of the interface oriented towards WLAN)
  InnerInterfaceIP =

  (Default value is 67)
  DHCPServerListenPort =

  (Default value is 67)
  DHCPClientThinkServerPort =

  (Name of the interface oriented towards WLAN)
  InnerInterfaceName =

Authentication providers

  Name = RTM (CHOOSE THIS!)
  URL = https://rtm.science.unitn.it/gate/auth/login.php
  Prefix = https://rtm.science.unitn.it/gate/auth/
  Domain = rtm.science.unitn.it
  IP = 193.205.194.21

  Name = Access
  URL =
  Prefix = http://192.168.0.2/gate/local/
  Domain = 192.168.0.2
  IP = 192.168.0.2

  Name =

SIP Plugin Parameters
```

```
  Enable SIP Plugin (0: No, 1: Yes) = 0
  Debugging mode (0: No, 1: Yes) = 0

SIP Servers

  Name =

CAPWAP Parameters:
  Enable CAPWAP Plugin (0: No, 1: Yes) = 0
  Path of parameters file =
  IP address of configuration service =
  UDP port of configuration service =

Perpetually Authorized clients

  Name = Forever_Authorized_User
  Email = Forever_Authorized_User@mail_domain
  MAC = 01:12:23:34:5A:BC
  IP = 192.168.12.10

  Name =
```

Notes

For a right routing of the packets, you must add some information to route the packet:

- add a rule about the default gateway for the external interface (eth0)
  `route add default gw 193.205.194.1`
- add a rule about the private IP address (for interface eth1)
  `route add -net 192.168.12.0 netmask 255.255.255.0`

For further information about parameters of the configuration, and how to configure proxy server, web server, please refer to 6.6.

6.4 Type IV configuration

The reference figure in this configuration is Figure 6.1. In this case the *Server1* contains all the required software: Gateway, Gatekeeper, and webserver.

The proxy server is required if the IP address used for wireless network cannot be managed by the external LAN.

The IP address in this example are:

- **192.168.10.2** the IP address of **eth0** interface of **Server 1**
- **172.31.194.3** the IP address of **eth1** interface of **Server 1**
- **172.31.194.0** the IP address class for wireless LAN

This configuration is an operative implementation of the authentication system. The IP address of eth0 interface is not public: it is a private static IP address because the system is running in a private network that is able to route it. Besides, the class of IP

address uses for the WLAN is managed by the External LAN, so in this configuration the involved proxy server is the proxy server in the External LAN.

6.4.1 How to configure the system

Gateway

The configuration file for Gateway can be found in the *Example* directory: "configuration1.gw". This file must be saved in the *parameter* to force the system to read it during the start-up.

```
configuration1.gw
Gateway configuration for the actual faculty server.

Inner interface network = 172.31.194.0
Inner interface netmask = 255.255.255.0
Inner interface ARP domain = 172.31.194.1
Inner interface ARP mask = 255.255.255.255
Inner interface properties name = eth1
Inner interface capture name = eth1

Outer interface network = 0.0.0.0
Outer interface netmask = 0.0.0.0
Outer interface ARP domain = 172.31.194.0
Outer interface ARP mask = 255.255.255.0
Outer interface properties name = eth0
Outer interface capture name = eth0
Outer interface gateway = 192.168.10.1

Tunnel IP (for gatekeeper communications) = 127.0.0.1
Tunnel listening UDP port = 54270
Command interface listening IP = 127.0.0.1
Command interface listening TCP port = 54270
Gatekeeper's tunnel IP = 127.0.0.1
Gatekeeper tunnel listening port = 54271
Gatekeeper's reporter IP = 192.168.10.2
Gatekeeper reporter listening port = 54272

Dropped packet sink IP = 0.0.0.0
Dropped packet sink UDP port (0 if none) = 0

DNS servers (semicolon-separated list)
     = 193.205.194.23;192.168.121.2;192.168.121.3
Initial HTTP server = 172.31.194.3
Proxy servers (semicolon-separated list)
     = 193.205.213.166;193.205.206.25
RADIUS server = 192.168.194.168

Submit all SIP packets to Gatekeeper?
(0: No, 1: Yes) = 0
```

```
Run mode (0: console; 1: daemon) = 1
Log file (only for daemon) = gateway.log
Transparent mode = 0
PID file = /var/run/gateway.pid
```

Gatekeeper

The configuration file for Gatekeeper can be found in the *Example* directory: "configuration1.gk". This file must be saved in the *parameter* to force the system to read it during the start-up.

```
configuration1.gk
Gatekeeper configuration for the actual faculty server.

Gatekeeper IP address configuration

  Authorization IP = 192.168.10.2
  Authorization port = 54273
  Authorization relay IP
  (same as above if relay not present) = 193.205.213.112

  Report IP = 192.168.10.2
  Report port = 54272

  Dispatcher IP
  (gatekeeper's tunnel endpoint) = 127.0.0.1
  Dispatcher listening port = 54271

Gateway IP address configuration

  Gateway tunnel IP for gatekeeper communications
    (gateway's tunnel endpoint) = 127.0.0.1
  Gateway tunnel UDP port = 54270
  Gateway command interfaceIP = 127.0.0.1
  Gateway command TCP port = 54270

DHCP information

  DHCP class C network IP = 172.31.194.0
  DHCP first assignable IP = 172.31.194.51
  DHCP last assignable IP = 172.31.194.253
  IP of DNS server = 193.205.194.23
  IP of gateway (seen by wireless hosts) = 172.31.194.1
  IP network mask = 255.255.255.0
  IP network domain = science.unitn.it
  Fake address of the DHCP server = 192.168.10.2
  Short lease time (for unauthorized clients) = 180
  Long lease time (for authorized clients) = 1800
```

```
Capture information

  redirection URL
    = http://192.168.10.2/gate/local/choose.php
  redirection URL when no proxy is set
    = http://172.31.194.3/gate/local/noproxy.php
  Proxy configuration URL
    = http://172.31.194.3/gate/local/wpad.dat
  Authorization renewal time = 600

Files
  Run mode (0: console; 1: daemon) = 1
  Status table dump file = gatekeeper.dat
  Program log file = gatekeeper.log
  PID file = /var/run/gatekeeper.pid

DHCP Relay Information

  RelayActivated = 0

  DHCPServerIP = 193.205.194.23

  OuterInterfaceIP = 192.168.10.2

  (Must be equal to OuterInterfaceIP when DHCP doesn't pass throught NAT)
  NatPublicIP = 192.168.10.2

  (IP Address of the interface oriented towards WLAN)
  InnerInterfaceIP = 172.31.194.3

  (Default value is 67)
  DHCPServerListenPort = 67

  (Default value is 67)
  DHCPClientThinkServerPort = 67

  (Name of the interface oriented towards WLAN)
  InnerInterfaceName = eth1

Authentication providers

  Name = Pimpa (EXPERIMENTAL)
  URL
    = https://pimpa.science.unitn.it/gate/auth/login.php
  Prefix = https://pimpa.science.unitn.it/gate/auth/
  Domain = pimpa.science.unitn.it
  IP = 193.205.194.149

  Name = Access
  URL =
```

```
  Prefix = http://192.168.10.2/gate/local/
  Domain = 192.168.10.2
  IP = 192.168.10.2

  Name =

SIP Plugin Parameters

  Enable SIP Plugin (0: No, 1: Yes) = 0
  Debugging mode (0: No, 1: Yes) = 0

SIP Servers

  Name =

CAPWAP Parameters:
  Enable CAPWAP Plugin (0: No, 1: Yes) = 0
  Path of parameters file =
  IP address of configuration service =
  UDP port of configuration service =

Perpetually Authorized clients

  Name = Forever_Authorized_User_With_Dynamic_IP
  Email = Forever_Authorized_User@mail_domain
  MAC = 01:12:23:34:5A:BC
  IP = 0.0.0.0

  Name = Forever_Authorized_User_With_Static_IP
  Email = Forever_Authorized_User@mail_domain
  MAC = 01:12:23:34:5A:BC
  IP = 172.31.194.10

  Name =
```

Notes

In this configuration the class of IP addresses must be routed by the external network. If you don't have a IP class with this requirement, you need a second server between the *Server1* and the *External LAN*. The connection between the two server is a point-to-point connection (with IP address without routed requirement) and the private static IP address (that can be routed by the external network) is the IP address of the *Server2* interface towards the external network. This IP address con be a public one.

For a right routing of the packets, you must add some information to route the packet:

- add a rule about the default gateway for the external interface (eth0)
  `route add default gw 192.168.10.1`

In the list of **Perpetually Authorized clients** the authorized users with a dynamic IP address request the IP trhough DHCP request. While the authorized users with

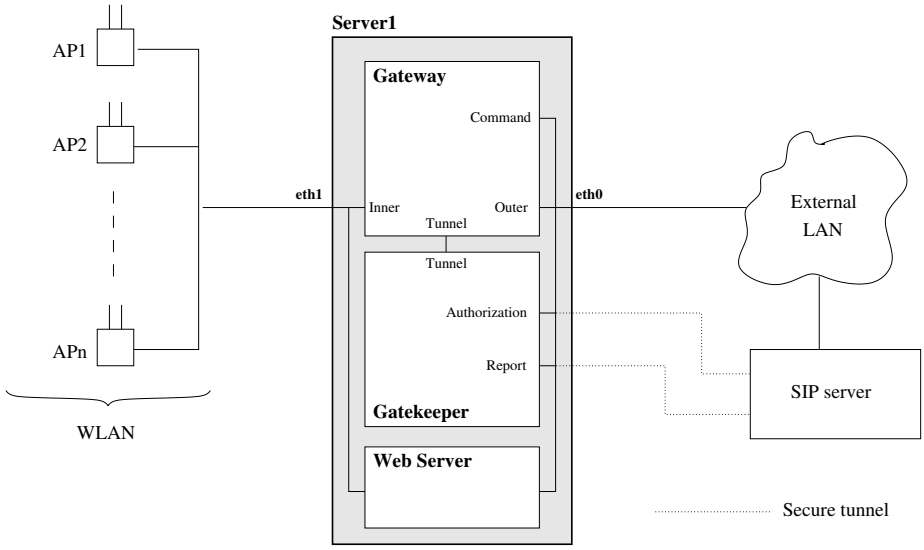


Figure 6.4: Network configuration for example 5

a static IP address does not requested it through DHCP transactions: it is manually configured on the user's machine. The configuration of DHCP server is not managed by the Uni-Fy system: in the server can be declarations of allowed address pool or can be static assignment related to MAC address for devices that are not able to dynamically request an IP address.

For further information about parameters of the configuration, and how to configure proxy server, web server, please refer to 6.6.

6.5 Type V configuration

In this case the *Server1* contains almost all the required software: Gateway, Gatekeeper, and webserver. SIP registrar is required in order to make SIP plug-in work and it can be either in the internal either in the external LAN. Also at least one Access Points equipped with OpenWRT firmware is required in order to make CAPWAP plug-in work properly.

The proxy server is required if the IP address used for wireless network cannot be managed by the external LAN.

The IP address in this example are:

- **192.168.90.1** the IP address of **default route** towards external LAN
- **192.168.90.3** the IP address of **eth0** interface of **Server 1**
- **192.168.90.130** the IP address of **eth1** interface of **Server 1**
- **192.168.90.128** the IP address class for wireless LAN
- **192.168.90.150** the IP address of **AP equipped with OpenWRT firmware**
- **193.205.213.21** the IP address of **SIP registrar** (e.g. SER)

This configuration is an operative implementation of both authentication system, HTTP and SIP. Moreover, AP are CAPWAP enabled. The IP address of eth0 interface is not public: it is a private static IP address because the system is running in a private network that is able to route it. Besides, the class of IP address uses for the WLAN is managed by the External LAN, so in this configuration the involved proxy server is the proxy server in the External LAN.

6.5.1 How to configure the system

Gateway

The configuration file for Gateway can be found in the *Example* directory: "configuration5.gw". This file must be saved in the *parameter* to force the system to read it during the start-up.

```
Inner interface network = 192.168.90.128
Inner interface netmask = 255.255.255.128
Inner interface ARP domain = 192.168.90.129
Inner interface ARP mask = 255.255.255.255
Inner interface properties name = eth1
Inner interface capture name = eth1

Outer interface network = 0.0.0.0
Outer interface netmask = 0.0.0.0
Outer interface ARP domain = 192.168.90.128
Outer interface ARP mask = 255.255.255.128
Outer interface properties name = eth0
Outer interface capture name = eth0
Outer interface gateway = 192.168.90.1

Tunnel IP (for gatekeeper communications) = 127.0.0.1
Tunnel listening UDP port = 54270
Command interface listening IP = 127.0.0.1
Command interface listening TCP port = 54270
Gatekeeper's tunnel IP = 127.0.0.1
Gatekeeper tunnel listening port = 54271
Gatekeeper's reporter IP = 192.168.90.3
Gatekeeper reporter listening port = 54272

Dropped packet sink IP = 0.0.0.0
Dropped packet sink UDP port (0 if none) = 0

DNS servers (semicolon-separated list) =
    193.205.194.23;192.168.121.2;192.168.121.3
Initial HTTP server = 192.168.90.130
Proxy servers (semicolon-separated list) =
    193.205.213.166;193.205.206.25
RADIUS server = 192.168.194.168

Submit all SIP packets to Gatekeeper?
```

```
(0: No, 1: Yes) = 0

Run mode (0: console; 1: daemon) = 1
Log file (only for daemon) = /home/Uni-Fy/gateway.log
Transparent mode = 0
PID file = /var/run/gateway.pid
```

Gatekeeper

The configuration file for Gatekeeper can be found in the *Example* directory: "configuration5.gk". This file must be saved in the *parameter* to force the system to read it during the start-up.

```
Gatekeeper IP address configuration

  Authorization IP = 192.168.90.3
  Authorization port = 54273
  Authorization relay IP
  (same as above if relay not present) = 192.168.90.3

  Report IP = 192.168.90.3
  Report port = 54272

  Dispatcher IP
  (gatekeeper's tunnel endpoint) =  127.0.0.1
  Dispatcher listening port = 54271

Gateway IP address configuration

  Gateway tunnel IP for gatekeeper communications
  (gateway's tunnel endpoint) = 127.0.0.1
  Gateway tunnel UDP port = 54270
  Gateway command interfaceIP = 127.0.0.1
  Gateway command TCP port = 54270

DHCP information

  DHCP network IP = 192.168.90.128
  DHCP first assignable IP = 192.168.90.200
  DHCP last assignable IP = 192.168.90.210
  IP of DNS server (0.0.0.0 if using relay) = 193.205.194.23
  IP of gateway (seen by wireless hosts) = 192.168.90.129
  IP network mask = 255.255.255.128
  IP network domain = 192.168.90.128
  Fake address of the DHCP server = 192.168.90.3

  Short lease time (for unauthorized clients) = 180
  Long lease time (for authorized clients) = 1800
```

```
Capture information

  redirection URL
     = http://192.168.90.3/gate/local/choose.php
  redirection URL when no proxy is set
     = http://192.168.90.3/gate/local/choose.php
  Proxy configuration URL  =
  Authorization renewal time = 600

Files

  Run mode (0: console; 1: daemon) = 1
  Status table dump file = /home/Uni-Fy/gatekeeper.dat
  Program log file = /home/Uni-Fy/gatekeeper.log
  PID file = /var/run/gatekeeper.pid

DHCP Relay Information (Experimental)

  RelayActivated = 0
  DHCPServerIP = 127.0.0.1
  OuterInterfaceIP = 127.0.0.1

  NatPublicIP =
  InnerInterfaceIP =
  DHCPServerListenPort =
  DHCPClientThinkServerPort =
  InnerInterfaceName = eth1

Authentication providers

  Name = PIMPA
  URL = https://pimpa.science.unitn.it/gate/auth/credentials.php
  Prefix = https://pimpa.science.unitn.it/gate/auth/
  Domain = pimpa.science.unitn.it
  IP = 193.205.194.149

  Name = Access
  URL =
  Prefix = http://192.168.90.3/gate/local/
  Domain = 192.168.90.3
  IP = 192.168.90.3

  Name =

SIP Plugin Parameters

  Enable SIP Plugin (0: No, 1: Yes) = 1
  Debugging mode (0: No, 1: Yes) = 0

SIP Servers
```

```
  Name = SER
  IP =  193.205.213.21
  Authentication mode (0:passive, 1: active) = 1

  Name =

CAPWAP Parameters:
  Enable CAPWAP Plugin (0: No, 1: Yes) = 1
  Path of parameters file
      = /home/Uni-Fy/parameters/template1.cw
  IP address of configuration service = 127.0.0.1
  UDP port of configuration service = 54282

Perpetually Authorized clients

  Name =
```

Notes

In this configuration the class of IP addresses must be routed by the external network. If you don't have a IP class with this requirement, you need a second server between the *Server1* and the *External LAN*. The connection between the two server is a point-to-point connection (with IP address without routed requirement) and the private static IP address (that can be routed by the external network) is the IP address of the *Server2* interface towards the external network. This IP address can be a public one. Note also that in this example only 10 clients can be attached to the WLAN, but in principle, since the configuration of IP address of this example, there could be up to 125 clients. It is sufficient to change *DHCP first assignable IP* and *DHCP last assignable IP* parameters.

For a right routing of the packets, you must add some information to route the packet:

- add a rule about the default gateway for the external interface (eth0)
  `route add default gw 192.168.90.1`

For further information about parameters of the configuration, and how to configure proxy server, web server and SIP servers please refer to 6.6.

6.6 General information

In this section the most important parameters of configuration file for Gateway and Gatekeeper are analyzed and a brief introduction how to configure the needed software is shown. For software like proxy- or web-server only the configurations for this network are shown. For further information or how to install them, please refer to the manual of the software.

Parameters of Gateway

- **Inner interface ARP domain** must respond to queries from users that have a private IP address in the WLAN. The same physical interface eth1 of *Server1*

has one physical IP address and one (or more) fake IP address(es) (as gateway and/o DHCP server).

- **Outer interface network** specifies what the system sense from this interface (eth0): *0.0.0.0* means "everything".
- **Outer interface ARP domain** specifies which of ARP queries (from external network) the system must answer.
- **Outer interface gateway** is the next hop for the packets that must be forwarded to the external LAN (it can be the IP address of the gateway of the External LAN).
- **DNS servers**: it can be an IP address of a real DNS server in the External LAN or the IP address of the interface of the server connected to the WLAN.

Parameters of Gatekeeper

- **IP of DNS server** is the IP of DNs server that is notified to the user in the DHCP transaction.
- **Fake address of the DHCP server** if the IP address ".1" is used, the ARP response are warranted
- **redirection URL** is the machine where the web server is. The URLs for the redirection use the alias set in the web server configuration.
- **redirection URL when no proxy is set** if the proxy is not set the web server must be reached through the interface of the Gateway in the wireless LAN.
- **Authentication providers**: if you set a URL in the parameters, the provider is a real remote authentication provider and its name (and link) is in the list of possible providers. Otherwise if you don't set a URL, the provider is the local provider that has only the page for the choice (choose.php) among the possible providers: this URL must be reachable by the user also if it is not an authentication server. Providers with URL parameter are mentioned in the choose.php (if they are more than one), while providers without URL parameter are not included.
- **Perpetually Authorized clients**. There are two way to set a static IP address: with or without description of the parameter. Also AP can be considered clients if they have a static IP address. These parameters are read from the Gatekeeper during the start-up.

Proxy server

We use *Squid* and we describe briefly how to set the basic parameters. For this purpose you must set ACL rules for your requirements

```
acl wireless_src src 192.168.12.0/24
acl server_src 192.168.0.2/32
http_access allow wireless_src
http_access allow server_src
```

NB: you must insert these commands before the line with `http_access deny all`.

Web server

You must define an alias:
`Alias /gate/ ''/root/wilmaproject/gateway/web''`
We use *Apache 2* and the alias can be inserted in a file in the *conf.d* directory: the files in this directory are additive configuration that are included in *apache2.conf* file.

DNS server

If the external network has its own DNS server and it can be reached by the interface with public IP of the Server2, a DNS cache is not required in the Server2. Otherwise you must install a DNS cache in the Server2.

SIP servers

In this subsection we will show how to install and configure a *PASSIVE* and an *ACTIVE* SIP server.

- **PASSIVE server** A PASSIVE server is absolutely transparent from SIP server point of view, it should only run in the standard way, without any customization.
- **ACTIVE server**

  An ACTIVE server instead, requires some modification on the server side. In our case, we've chosen as SIP server the SIP Express Router (SER), so an example of how to make it work is explained here.

  The *unify module* we've created (and that can be found in the `SER` directory), and allows SER communicating with the Uni-Fy gates and provides a SIP authentication method.

  1. download SER (version 0.9.6 possibly) source code from the http://www.iptel.org/ser/;
  2. create the *unify* directory in the subdirectory *modules*;
  3. enter in the unify directory and create the *Makefile* with the following lines

     ```
     #
     # WARNING: do not run this directly,
             # it should be run by the master Makefile

     include ../../Makefile.defs
     auto_gen=
     NAME=unify.so
     LIBS=-lrt

     include ../../Makefile.modules
     ```

     These lines are used to compile the Unify module; this module used the shared memory then we have to compile it with the *real time* library (-lrt);
  4. copy *unify_mod.c, unify_macro.h, net_funcs.h, sem_funcs.h, shm_funcs.h, net_funcs.c, sem_funcs.c, shm_funcs.c* in the directory just created;
  5. exec the *make modules* command in the root SER directory;
  6. copy *unify.so* file from *ser/modules/unify/* to */usr/lib/ser/modules*;
  7. restart the SER proxy.

Configuration of OpenWRT on AP

In order to use CAPWAP plug-in, AP must be updated with OpenWRT firmware plus wtp program, included in Uni-Fysources. For complete this task you can refers to official documentaion of OpenWRT. In order to run wtp program, you must connect with SSH session to AP, then move in directory /etc/sbin and digit ./wtp CAPWAP.pmt. CAPWAP.pmt contains essential parameters of CAPWAP protocol described below:

- **In UDP Port**— UDP Port for incoming packets, default value 54280.
- **Out UDP Port**— UDP Port for outcoming packets, default value 54281.
- **Netmask**— The netmask that AP must use, it can be 0.0.0.0.
- **Gateway** — IP address of AP gateway, it can be 0.0.0.0.
- **AC IP** — IP Address of AC (must be the same of Uni-Fy Gatekeeper)
- **AC Name** — A nickname of AC, useful in WLAN with multiple AC.
- **WTP IP** — IP address of AP.
- **WTP Name** —A nickname of AP, used only if there isn't wtp preconfiguration on AC.
- **WTP Location** — Location of AP, used only if there isn't wtp preconfiguration on AC.
- **Default Channel** — Default channel of WTP, used only if there isn't wtp preconfiguration on AC.
- **Default tx power (dBm)** — Default tx power (in dBm) of WTP, used only if ther isn't wtp preconfiguration on AC.
- **Default ESSID** — Default ESSID of WTP, used only if there isn't wtp preconfiguration on AC.
- **Max Discoveries** — Max number of Discovery Request that WTP is able to send before enter in sulking state (Silent Interval), default value 3.
- **Max Retransmission** — Max number of retransmission that WTP is able to send before enter in sulking state (Silent Interval), default value 3.
- **Statistics Timer** — Statistics Timer in seconds, default value 120.
- **Discovery Interval** — Discovery Interval, default value 5.
- **Echo Interval** — Echo Interval, default value 30.
- **Response Timeout** —Response Timeout, default value 1.
- **Silent Interval** — Silent Interval1, default value 20.
- **Wait Join** — Wait Join, default value 15.

Chapter 7

Linking to advanced kernel features

Virtual LAN (VLAN) management and Network Address Translation (NAT) functionalities are common features found in many Layer-2 and -3 networking devices. These features are not implemented in the Uni-Fy system, but are easily compiled in the Linux kernel. In this appendix we describe the steps to be taken in order to obtain a working Uni-Fy system with these additional features.

7.1 Virtual LAN management

If the Uni-Fy system is placed on an Ethernet line that operates on different VLANs in trunking mode, the physical interface will receive Ethernet packets in the 802.1Q format. However, the Uni-Fy code does not recognize it.

Virtual LAN interfaces can be accessed by creating logical Ethernet sub-interfaces of the physical devices. For instance, on `eth0` a new virtual device `eth0.70` defines the device that only manages 802.1Q packets tagged as VLAN 70. Many utilities can be used to create such devices.

Figure 7.1 shows a very simple scenario where only one physical LAN exists and connects both the wireless part (one or more APs), the Uni-Fy server and the border gateway, with the Gateway machine being provided with just one Ethernet port. Provided that the Uni-Fy Linux kernel is compiled vith VLAN support, and that the AP supports the 802.1Q standard, the Gateway functionality is obtained by bridging between the two logical LANs.

7.2 Network Address Translation

The Network Address Translation (NAT) functionality can be performed by the kernel's `netfilter` mechanism that can be controlled via the `iptables` commands. To concatenate the Uni-Fy system and thee NAT kernel functions on the same machine, we make use of internal virtual interfaces (called TUN/TAP).

For this reason, the following prerequisites must be satisfied:

- TUN/TAP functionality compiled in the kernel

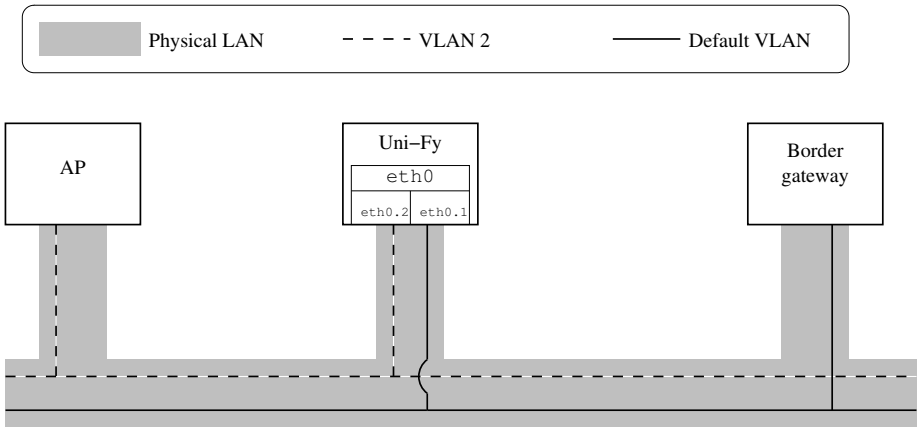


Figure 7.1: A small network based on Virtual LANs.

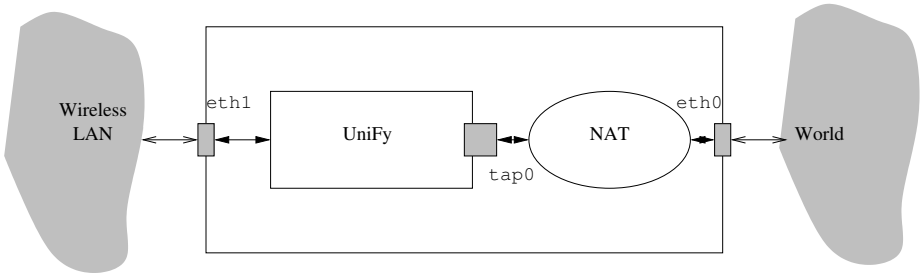


Figure 7.2: Uni-Fy-NAT chaining.

- Routing functionality compiled in the kernel
- `netfilter` and `iptables` packages
- `tunctl` utility present (often available in User-Mode Linux packages).

In the following example, we shall consider this network configuration:

- Inner interface: eth1.
- Inner network: 10.0.0.0/24
- Outer interface: eth0
- Outer IP: 193.205.193.87 (a class C public address)

The basic idea is to create an internal (virtual) network interface, known as `tap0` (also used in many other contexts such as User Mode Linux or VPN tools), set the Uni-Fy program to work between `eth1` and `tap0` (which is seen by the software as a character device, rather than a network interface), and put the rest of the machine (iptables and the regular TCP/IP stack) to work between `tap0` and `eth0`.

1. Configure the inner interface. Its IP address shall not be used, since we are only interested in layer 2 communications (the rest is fully managed by the Wilma-Gate), so a dummy address in any private unused network is OK. An IP address is mandatory, however:

   ```
   ifconfig eth1 192.168.200.1 up
   ```

2. Create the TUN/TAP device node (character device, major 10, minor 200)

   ```
   mkdir /dev/net
   mknod /dev/net/tun c 10 200
   ```

3. Create the tap0 interface:

   ```
   tunctl -t tap0
   ```

4. Configure the tap0 interface. Its address shall be used as the web server address (so it should appear on the Gatekeeper configuration file); the best choice is to devise another private address. The tap0 interface shall be used by the system to output all traffic directed to the wireless network, so the appropriate routing rule must be created:

   ```
   ifconfig tap0 192.168.123.2 up
   route add -net 10.0.0.0/24 tap0
   ```

5. Set up kernel routing:

   ```
   echo 1 > /proc/sys/net/ipv4/ip_forward
   ```

6. Reset iptables:

```
iptables -F
iptables -t nat -F
```

7. Since we are going to use `eth1` just for the Uni-Fy gateway, block kernel forwarding from inner network. It is also feasible to disable the INPUT and OUTPUT chains, in order to be sure that no packet coming from regular TCP/IP applications will ever cross that interface:

   ```
   iptables -A FORWARD -i eth1 -j DROP
   iptables -A FORWARD -o eth1 -j DROP
   ```

8. Setup NAT towards `eth0` (note that the other endpoint is necessarily `tap0`, because `eth1` cannot forward anymore):

   ```
   iptables -t nat -A POSTROUTING -s 10.0.0.0/24 \
                          -o eth0 -j MASQUERADE
   ```

9. Tune the configuration files and run the system.

The initial lines of the gateway configuration file in our example are the following.

```
Inner interface network = 10.0.0.0
Inner interface netmask = 255.255.255.0
Inner interface ARP domain = 10.0.0.1
Inner interface ARP mask = 255.255.255.255
Inner interface properties name = eth1
Inner interface capture name = eth1

Outer interface network = 0.0.0.0
Outer interface netmask = 0.0.0.0
Outer interface ARP domain = 10.0.0.0
Outer interface ARP mask = 255.255.255.0
Outer interface properties name = tap0
Outer interface capture name = tap0
Outer interface gateway = 192.168.123.2
```

When an interface is called `tap`n, the system automatically manages it via the proper system hooks rather than the libpcap mechanism.

Appendix A

List of changes

The following lists keep the evolution of the system and the release of the document.

A.1 System Evolution

wilmagate-20050617 Last stable version of WilmaGate system. Starting point for evolution of Uni-Fy.

unify-20060421 Last version of Uni-Fy with minor changes and bugfix.

unify-20060729 Uni-Fy with new feature about management of IP addresses: implementation of DHCP Relay agent to provide use of external DHCP server.

unify-20070601 Uni-Fy with new features: support for SIP-based authentication and for CAPWAP protocol for network management. Furthermore new php pages user management and minor bugs fix.

A.2 Specification Document

Ver. 1.0 Related to WilmaGate system description.

Ver. 1.1 Related to unify-20060729 version. Description of new feature about management of IP addresses.

Ver. 1.2 Related to unify-20070727 version. Description of new SIP and CAPWAP features

Appendix B

Licence of Uni-Fy

Developed by TWELVE staff, University of Trento, Italy.

University of Trento
DIT - Department of Informatics and Telecommunications
TWELVE Prin Project
`http://twelve.dit.unitn.it/`

This file is part of the Uni-Fy system, created as part of the TWELVE Project, supported by Italian Ministry for University and Research (MIUR). More info at

http://twelve.dit.unitn.org/

To download the project:

- in tarball format (old release)
  http://twelve.dit.unitn.it/
- in tarball format (new release)
  http://networking.dit.unitn.it/

The Uni-Fy Licence of Use

- The Uni-Fy shall not be sub-licenced either commercially or free of charge, as is or in part or as a part of any other package.
- The licensee agrees to properly credit Uni-Fy and its copyright owners in any technical, scientific or divulgative publication, whatever publishing means are used, that is based entirely or in part on Uni-Fy use, as well as in any booklet, web-site or brochure that advertise services based on WilmaGate use.
- If the Uni-Fy is modified, the copyright owner shall be notified.

THE Uni-Fy IS PROVIDED ”AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Appendix C

Licence of WilmaGate

Developed by NetMob Group, University of Trento, Italy.

University of Trento
DIT - Department of Informatics and Telecommunications
Computer Networks and Mobility Research Program
http://netmob.unitn.it/

This file is part of the WilmaGate system, created as part of the WILMA Project, sponsored by the Autonomous Province of Trento and participated by ITC-irst, the University of Trento and Alpikom Spa. More info at

http://www.wilmaproject.org/

To download the project:

- in tarball format
  http://netmob.unitn.it/wilmagate.html
- connect to CVS repository
  :pserver:cvs@rtm.unitn.it:/home/cvsroot

In both previous cases, the download is password protected. To obtain the password you must send a mail to

wilmagate@dit.unitn.it

The WilmaGate Licence of Use

- The above copyright notice and this permission notice shall be included in all copies or substantial portions of the WilmaGate.
- The WilmaGate shall not be sub-licenced either commercially or free of charge, as is or in part or as a part of any other package.
- The licensee agrees to properly credit WilmaGate and its copyright owners in any technical, scientific or divulgative publication, whatever publishing means are used, that is based entirely or in part on WilmaGate use, as well as in any booklet, web-site or brochure that advertise services based on WilmaGate use.
- If the WilmaGate is modified, the copyright owner shall be notified.

THE WilmaGate IS PROVIDED ”AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.