low Energy COnsumption NETworks

DELIVERABLE D5.5

PROTOCOL EXTENSIONS

Grant Agreement Number:	**258454**
Project Acronym:	**ECONET**
Project Title:	**low Energy COnsumption NETworks**
Funding Scheme:	Collaborative Project
Starting Date of the Project:	01/10/2010 *dd/mm/yyyy*
Duration:	36 months (original), 39 months (amendment request)
Project Coordinator:	Name: Raffaele Bolla Phone: +39 010 353 2075 Fax: +39 010 353 2154 e-mail: raffaele.bolla@unige.it

Due Date of Delivery:	M33 *Mx* (30/06/2013 *dd/mm/yyyy*)
Actual Date of Delivery:	04/07/2013 *dd/mm/yyyy*
Workpackage:	**WP5 – *Green Strategies at the Control Plane***
Nature of the Deliverable:	R
Dissemination level:	PU
Editors:	ALU, CNIT, MLX, TEI, NVR
Version:	1.0

Disclaimer

Copyright

Table of Contents

DISCLAIMER **2**

COPYRIGHT **2**

TABLE OF CONTENTS **3**

1 EXECUTIVE SUMMARY **5**

2 INTRODUCTION **6**

3 CONTROL PLANE REFERENCE ARCHITECTURE **7**

3.1 OSPF-TE PROTOCOL EXTENSION 8
3.2 INTERFACING OPTIMIZATION ALGORITHMS 10

4 GAL OBJECTS TRANSPORT PROTOCOL FOR NETWORK POWER MANAGEMENT **12**

4.1 GAL BINDINGS FOR XML OVER HTTP REST 13
4.1.1 Place of the GAL interface within the ECONET management framework *13*
4.1.2 Introduction to the REST-style interfaces *14*
4.1.3 Further network API bindings for the GAL *15*
4.1.4 The GAL REST XML Tree *15*
4.1.5 GAL operations *17*
4.1.6 Distributed GAL access with REST API - concept *20*
4.2 GAL BINDINGS FOR CORBA / TMF-814 (SECTION 2) 21
4.2.1 Architectural overview *21*
4.2.2 SPO1400 energy aware *22*
4.2.3 NBI TMF814 Impacts *23*
4.3 GAL BINDINGS FOR TR-069 27
4.3.1 Introduction *27*
4.3.2 Considerations *27*
4.3.3 Solution architecture *27*
4.3.4 Description of the Power Data Model *28*
4.3.5 Collection of History items *29*

5 GREEN EXTENSION OF OPENFLOW **30**

5.1 INTRODUCTION 30
5.2 NETWORK PROCESSOR ARCHITECTURE 32
5.3 THE OPENFLOW PROTOCOL EXTENSION TOWARDS GREEN CAPABILITIES 33
5.4 IN-NETWORK CONSOLIDATION 35
5.5 THE IN-NETWORK CONSOLIDATION ALGORITHM 35
5.6 TEST RESULTS 37

6 NETWORK CONNECTIVITY PROXY **40**

6.1 OVERVIEW 40
6.2 NCP SERVICE 41
6.2.1 Description *41*
6.2.2 Supported Actions *42*

6.2.3 Integration of UPnP Actions & NCP Rules *44*
6.3 LOWPOWER SERVICE 45
6.3.1 Description *45*
6.3.2 State Variables *46*
6.3.3 Supported Actions *46*
6.4 CONCLUSIONS AND FURTHER WORK 47
REFERENCES **48**

1 Executive Summary

Deliverable D5.5 “Protocol Extensions” reports all ECONET developments related to the extension of networking protocols with green capabilities. Where possible, this choice has been extensively used to support seamless network evolution and to take advantage from protocol flexibility (e.g., unused fields/valued).

Four main areas have been addressed:

Control Plane extension to carry Power Object information across the network, according to the ECONET GAL syntax, being also able to easily integrate optimization algorithms, as reported in D5.3. The proposal is - generally speaking - simple, but concretely effective to make the studied routing techniques applicable.

Network Power Monitoring framework, developing adaptation layers between the standard GAL interface, defined within the ECONET project, and common management interfaces such as Rest/XML, CORBA/TMF 814 and Broadband Forum TR-069/CWMP. This development allows interfacing standard available network nodes that have been later on integrated in the ECONET demo setup. This work is essential for extending the usage of the GAL interface outside a single physical object and then for exploiting its functionalities in complex distributed equipment and at the network level. These proposed extensions involve the most used management protocols and they contribute to make the GAL adoption a feasible task in a very large number of situations and scenarios.

Green extension of Openflow, enabling the usage of the GAL interface and methods into the emerging SDN paradigm. This activity has a twofold value: from one side, it projects the GAL in a very hot new network paradigm, SDN, effectively enabling its usage in future advanced equipment based on this approach; on the other side, it defines a very innovate way to extend the current Openflow protocol and architecture to handle the power management capabilities of the SDN devices. This last point might be a focal one in the near future to make the innovative developments in this area scalable.

Finally, the Network Connectivity Proxy protocol has been introduced as a key asset of ECONET technology. It is based on a suitable **extension of the UPnP** standard protocol. Here again, an important work that, if concretely adopted, contributes in a valuable manner to make NCPs definitely easier to introduce in most of the SOHO contexts.

Protocol syntax and additional details are provided in a dedicated Annex available as a separate document.

2 Introduction

In the following Sections we address the key topics related to the network control protocol extensions required for the support of the network-wide green functionality designed in the scope of the ECONET project.

During the execution of the project, the ECONET team has developed a variety of power saving/power management technologies for network devices in the data plane [5].

Such technologies can be grouped in three main categories [1]:

a) *Energy-saving solutions working inside the network device*, i.e., any hardware technology that exploits Adaptive Rate (AR), Low Power Idle (LPI), and Standby primitives. These approaches usually tend to be almost completely transparent to the networking protocol stack, since they are constrained inside the single device.

b) *Energy-saving solutions working between two devices*: this category groups all the approaches that can be adopted to save energy in network devices interconnected among themselves. Green extensions for link protocols belong to this category, which may include also mechanisms/frameworks to trigger wake-up signals and to shape the profiles of transmitted traffic, to optimize the behaviour or the “induced” energy consumption of directly attached nodes.

c) *Energy-saving solutions that bypass devices or functionalities*: this kind of approaches can be used at different network layers in order to avoid potential drawbacks caused by the use of standby primitives. Bypass approaches can be adopted for maintaining network functionalities and services running, when some network devices (or some of their components) enter standby mode. In fact, these approaches may allow migrating traffic transmission and reception, network protocol execution, and information processing towards other devices/components.

The exploitation of the energy saving techniques that involve two or more network devices (as per the solutions (b) and (c) described above) requires the involved devices to exchange information relevant to their power states and power management capabilities among themselves and towards the network Control Plane. With respect to such issue, in the scope of WP4 activities the ECONET Project has designed a standard and general purpose interface, called Green Abstraction Layer - GAL (see [1-6]), for exposing the novel green capabilities and functionalities towards control and monitoring frameworks.

The first aim of this deliverable is defining the needed extensions of current network Control and Management Plane protocols in order to carry the new data modelled by the GAL, so as to allow power network optimization and, more generally, power management. In this respect, the next Sections address the following key topics:

- a suitable Open Shortest Path First – Traffic Engineering (OSPF-TE) protocol extension to carry GAL objects for a generic Generalized Multi-Protocol Label Switching (GMPLS) [10] network controlled by Path Computation Client/Path Computation Element (PCC/PCE);
- GAL interfacing / adaptation to different network management protocols to facilitate the usage of the GAL technology in real networks;

- a green extension of the Openflow standard protocol, in accordance to the envisaged migration towards Network Functions Virtualization (NFV) and Software Defined Network (SDN) paradigms.

Besides the items listed above, this deliverable also includes the specification for a further control protocol extension needed to support a specific energy saving solution, based on the "*device bypass*" technique, developed in the scope of WP3: the Network Connection Proxy (NCP) [1]. The NCP function covers devices for network-related tasks whenever they fall in low-power states, so as to replace them in keeping their established connections active. The NCP solution designed in the scope of the Project has been focused on applications in the Home Network scenario and requires an exchange of information between the NCP itself and the covered Home Network Hosts for different purposes (as, for instance, mutual discovery, service registration and notification of power state transition events). In respect to this, the ECONET team has defined a new profile of the widespread Universal Plug and Play (UPnP) protocol to support the NCP specific functionality.

3 Control Plane reference architecture

This section introduces a general overview of the Alcatel-Lucent Control Plane (CP) activities in implementing the ECONET extensions for control plane protocols. The ALU CP architecture is an evolution of the DRAGON (Dynamic Resource Allocation via GMPLS Optical Networks) suite.

DRAGON is an open source GMPLS control plane that enables dynamic provisioning of network resources on an inter-domain basis, across heterogeneous network technologies, opening up the creation of dynamic, deterministic, and manageable end-to-end network transport services for high-end applications. The DRAGON architecture utilizes GMPLS as the basic building block for network element control and provisioning. Currently, the DRAGON software architecture is not PCC/PCE standard compliant. For ECONET purposes, we extended the original DRAGON architecture in order to:

1. Create a standard PCC/PCE chain [8], communicating via the PCE Protocol (PCEP) [9]
2. Extend the OSPF-TE protocol (Link State Packets (LSP) and Traffic Engineering Database (TED)) in order to inject the green parameters coming from the data-plane (device state, power consumption, etc.) into the control plane;
3. Extend the PCEP protocol in order to guarantee interoperability with other ECONET PCE architectures.

The following Figure 1 shows the generic Alcatel-Lucent CP architecture and communication protocols involved.

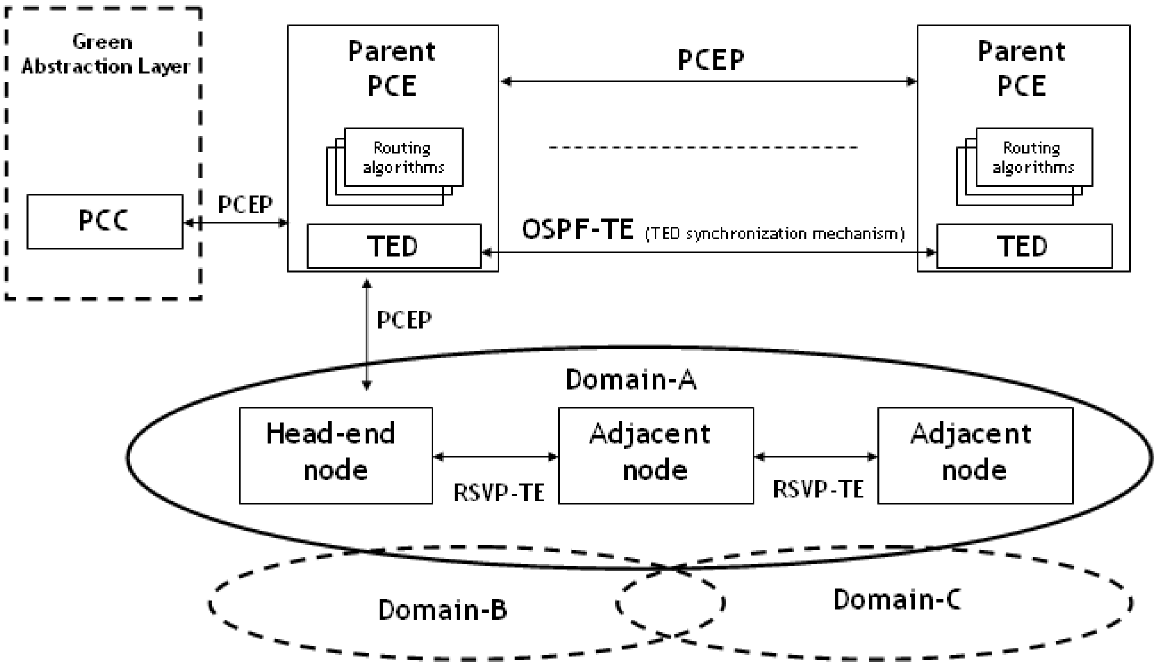


Figure 1: Alcatel-Lucent CP architecture.

The PCE architecture introduces a special computational entity that will cooperate with similar entities to compute the best possible path through multiple domains. A PCE is a node that has special path computation ability and receives path computation requests from entities known as path computation clients (PCCs). The PCE holds limited routing information from other domains, allowing it to possibly compute better and shorter inter-domain paths than those obtained using the traditional per-domain approach. Among other purposes, PCEs are also being advocated for CPU-intensive computations, minimal-cost-based TE-LSP placement, backup path computations, and bandwidth protection. Along with the process of identifying the requirements and development of the architecture accordingly, a plethora of work is underway at the PCE WG on the new communication protocols that will make this architecture work. This includes the development of new inter-PCE communication protocols and the introduction of extensions to existing underlying routing protocols. Request for Comments (RFC) 4655 [8] specifies a PCE-based architecture, while RFC 4657 covers PCE communication protocol generic requirements and, finally, RFC 4674 discusses the requirements for PCE discovery.

3.1 OSPF-TE protocol extension

We consider opaque Link State Advertisements (LSA) of the OSPF protocol for implementing the proposed GMPLS extensions. New Type, Length, Value (TLVs) are added to the Traffic Engineering (TE) extensions for OSPF. The TE-LSA format starts with the standard LSA header, with TLVs as payload, as shown in Figure 2. The detailed explanations of each field can be found in [10]. TE LSA defined two types of top-level TLVs. We propose to add sub-TLVs to link level TLV. Thus, the energy information is carried by setting up new sub-TLVs inside Link TLV (type 2) of TE LSAs. The Type field in sub-TLVs is considered in the range 32768-32777, as defined for experimental use. Length could be defined as 4 octets, as most of the other sub-TLVs defined in this level. Value is defined to carry the energy information.

It is composed of the following:

- The energy information, normally a value assigned to represents the power consumption [mW]
- The link status, composed by 12 power status values initially based on the EMAN IETF standard [11], is now based on the GAL syntax.
- The transition time matrix (12x12).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            LS age             |    Options    |      10       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       1       |                   Instance                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Advertising Router                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     LS sequence number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         LS checksum           |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Value...                            |
.                                                               .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2. TE-LSA format.

The flooding procedure follows the standard flooding procedure of OSPFv2. In the simulation model, no designated routers are selected. Thus, LSAs are flooded in broadcasting manner. *MinLSInterval* defines the minimum time to update the same entry in the Link State Database of each node, so that duplicated LSAs received from multiple neighbours can be ignored. After receiving a new LSA, the node decides whether to forward the LSA or discard it according to the LSA type and the carried timestamp. Link State Database is updated upon receiving valid LSAs. According to the definition of sub-TLVs for TE LSA, the defined sub-TLVs may occur at most once. TE LSAs are flooded upon changes of contents. The triggers for originating new TE LSAs are implemented with fixed time-outs. For nodes that do not support proposed TLVs, the TLVs will be ignored.

Routing calculation is based on the standard Dijkstra algorithm used in the OSPF-TE protocol. The “cost” assigned in route calculations will be updated by the energy cost carried in TE LSAs. The proposed model defines two types of cost value. These values could represent cheaper energy sources (0) and more expensive energy consumption sources (1), which applies to both link and node cost. In the proposed model, both link and node energy costs should influence the routing decision. However, as Dijkstra’s algorithm considers only one single value per edge, the weight assigned to Djikstra’s algorithm should be revised to fit the need. Thus, each edge cost is represented by link cost + 0.5 * (source node cost) + 0.5 * (destination node cost). The whole route cost will be the sum of all the edges’ costs. In this way, both node energy cost and link energy cost are taken into consideration.

3.2 Interfacing Optimization algorithms

In order to prove the effectiveness of optimization algorithms in a real network we need to identify a clever way to integrate those algorithms in the network management path.

First, some preliminary considerations: depending on the network size and its dynamicity we need to determine a suitable frequency sampling value *Fs* taking care of traffic behaviour, time to transfer network status *Sn(t)* to the optimization algorithm, time to transfer and implement the new network status *Sn(t+1)*. It is clear that if the traffic varies faster than the control loop the optimization effort is useless.

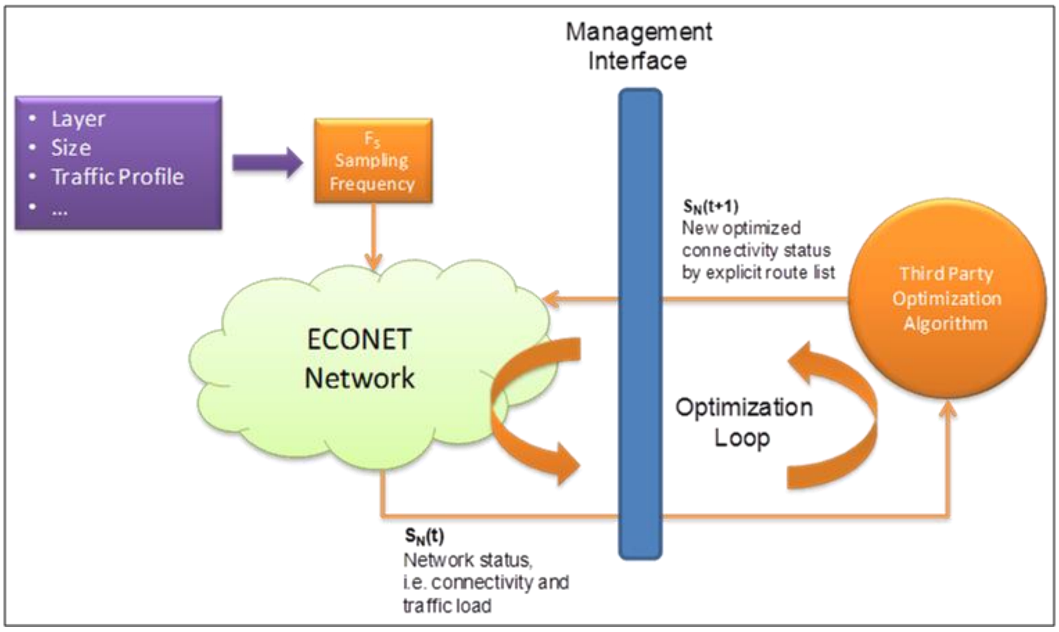

Management Interface
Layer
Size
Traffic Profile
...
F_S Sampling Frequency
ECONET Network
$S_N(t+1)$ New optimized connectivity status by explicit route list
Third Party Optimization Algorithm
Optimization Loop
$S_N(t)$ Network status, i.e. connectivity and traffic load

Figure 3. Integration of optimization algorithms in real networks

For large core networks, even if managed by a control plane, we need to apply optimization algorithms taking into account “long term” behaviour. The goal is mainly to switch off the unused resources, for instance in a night/day behaviour.

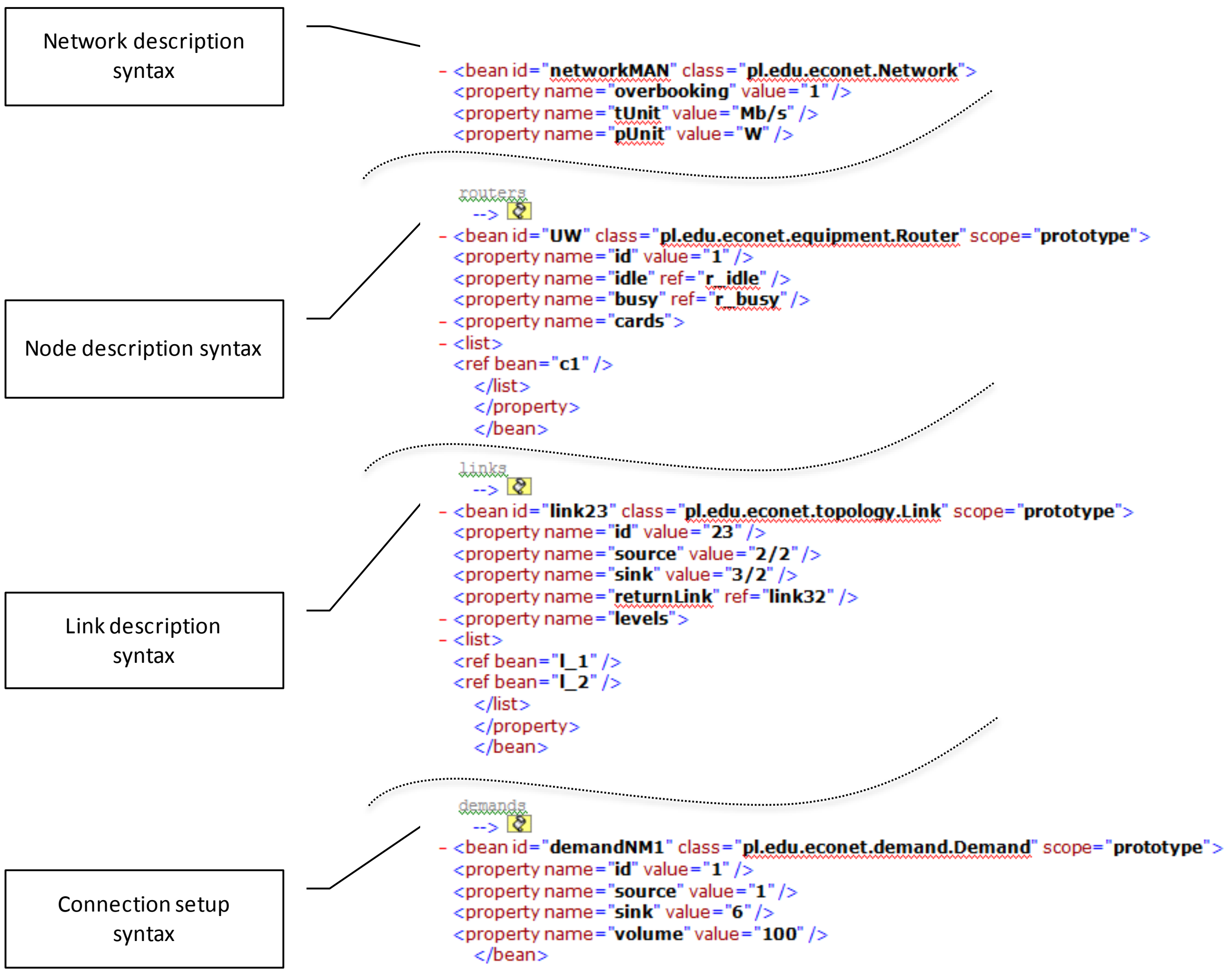
In the scenario of GMPLS networks, described in the previous Section, in order to avoid the complex integration process of optimization algorithms, we use the architecture described in Figure 3.

The status of the network, including topology and power consumption figures, is received by the optimization algorithm, which elaborates the new optimal (in terms of power consumption) configuration. The new configuration is then implemented by using the standard management interface. In the specific GMPLS case the explicit route is passed to the control plane to be implemented and the unused resources are switched off by using the GAL interface.

In order to verify the proposed approach, the NASK optimization algorithms has been selected and applied to the day/night scenario also used in the final demo of the project. A suitable interface (XML based) was defined to exchange information between the control plane and the optimization algorithm. The following “snapshot” (see Figure 4) is providing an example of the used syntax to pass network information and path setup request to the optimization algorithm. It returns the list of links used to implement the path by using power figures as metric.

```
ls task                        // list of completed and ongoing optimization problems,
stop <task id>                 // stopping of ongoing task,
ls network                     // list of available networks,
ls router                      // list of available routers,
ls card                        // list of available cards,
ls link                        // list of available links,
ls demand                      // list of available traffic demands,
ls EAS                         // list of available Energy-Aware-States of routers, cards and links
solve -BB <network name>       // Branch and Bound method,
solve -H <network name>        // Heuristic method,
exit                           // shutdown
```

a) Information and commands to activate network optimization procedure.



```
- <bean id="networkMAN" class="pl.edu.econet.Network">
  <property name="overbooking" value="1" />
  <property name="tUnit" value="Mb/s" />
  <property name="pUnit" value="W" />

  routers
  -->
- <bean id="UW" class="pl.edu.econet.equipment.Router" scope="prototype">
  <property name="id" value="1" />
  <property name="idle" ref="r_idle" />
  <property name="busy" ref="r_busy" />
- <property name="cards">
- <list>
  <ref bean="c1" />
    </list>
    </property>
    </bean>

  links
  -->
- <bean id="link23" class="pl.edu.econet.topology.Link" scope="prototype">
  <property name="id" value="23" />
  <property name="source" value="2/2" />
  <property name="sink" value="3/2" />
  <property name="returnLink" ref="link32" />
- <property name="levels">
- <list>
  <ref bean="l_1" />
  <ref bean="l_2" />
    </list>
    </property>
    </bean>

  demands
  -->
- <bean id="demandNM1" class="pl.edu.econet.demand.Demand" scope="prototype">
  <property name="id" value="1" />
  <property name="source" value="1" />
  <property name="sink" value="6" />
  <property name="volume" value="100" />
    </bean>
```

b) Syntax description of the interface.

Figure 4. Interface description between Emulated Network and external optimization task.

4 GAL objects transport protocol for Network Power management

The Green Abstraction Layer (GAL) forms the counterpoint of the research carried out by the whole of the ECONET consortium. The GAL is a flexible and modular interface and architecture for enhancing telecom network elements with energy awareness and energy-optimized behaviour.

The GAL is also a hierarchical and orthogonal approach, usable not only for the low, intra-equipment, level of hardware components (like a device line card controlling the energy-aware behaviour of the contained ports), but also at the higher, system-related levels, where devices and even Element Managers (EMS) or Network Management Systems (NMS) are controlled and monitored by higher-level management entities.

Besides the similarities of the two scenarios, the two levels of operations also differ, at least in the two following main aspects:

1. The levels of communication (status queries and control commands) are significantly different: among the systems, rather than communicating direct port states, and port level energy operation modes, here the information is much more strategic in nature, allowing space for the extensive GAL capabilities of autonomous decisions and control at the device levels and below.
2. Moreover, in the high-level scenario, the controlled and controlling entities are not located in the same hardware system (device), but they are typically network-linked, most typically over an IP network.

While the GAL approach is completely reasonable (considering the amount of data exchanged or the state-of-the-art organization of telecommunication networks and management frameworks), this opens up another problem of transporting GAL commands (which are originally defined as generic library functions), over an IP network, considering all the efficiency, security and domain-isolation practices which are very common in telecommunications.

A few examples:

- In spite of a broadband connection, elements (e.g., home gateways) in a customer home cannot typically be accessed by the operator directly, due to privacy considerations.
- Some telecom equipment families (including some demo equipment to be used in WP6), do not allow a direct management by any external system, but an EMS is provided, which accepts instructions only over a certain protocol, like TMF-814 SNMP, Web Services, etc.

To make the GAL applicable for networks, areas and sets of systems characterized by these constraints, protocol extensions and concretizations need to be defined. These definitions can be considered as different “bindings” of the GAL concept for different protocol technologies, to make them applicable for different scenarios and a priori conditions, similar to the cases described above.

The purpose of this section is to give an overview and also to describe in detail the concepts and implementations of these protocol extensions/bindings, i.e. the GAL implementation for the following protocols / technologies:

1. GAL bindings for XML over HTTP REST (Section 4.1)
2. GAL bindings for Corba / TMF-814 (Section 4.2)
3. GAL bindings for TR-069 (Section 4.3)

4.1 GAL bindings for XML over HTTP REST

This section specifies the XML/HTTP REST API binding of the GAL Application Programming Interface described in D4.3. We consider that document as the conceptual definition of a generic network-, device- and sub-device-level power management interface and also a programming language-style API definition.

While D4.3 can be used in rather straightforward way to implement a server library, or write a GAL client code (e.g., in “C” language) as done in D4.4, it does not define any binding for network communication between GAL server (e.g., a GAL-aware server) and a GAL network client (e.g., a higher level controlling application).

Here we specify a binding for the GAL that is conceptually equivalent to the D4.3 interface, but it is realized by using a network protocol (rather than library calls within runtime) as “vehicle” of information exchange.

4.1.1 Place of the GAL interface within the ECONET management framework

Being focused on the energy-related information only (states and measurements), the GAL alone is not always expected to be used to monitor and manage a complex network of telecommunication service devices, but rather it is an extension of other interfaces used for the discovery and monitoring of networks from other perspectives (topology, general performance, faults, etc.)

This is similarly reflected in the ECONET demonstration architecture (see D6.4), where the GAL interface described here is one of three interfaces used for communication between the managers and the managed networks and elements (see Figure 5):

- The Topology interface is the principal mechanism for describing the networks, the devices (including their internals if necessary), and the interconnections between ports on the devices. In addition, the topology information will contain the definition of the GAL interface access points available for executing GAL commands (similarly, points of access to performance data are also described here). The Topology interface is thus the key for all information exchange between the managers and managed systems.
- The second interface is the GAL itself.
- The third interface is used to query data from the managed systems (networks, devices, etc.) that fall outside of the GAL scope, but that are necessary for demonstrating and evaluating the energy-aware behaviour of the devices, especially the impact of the energy states, and the consequential trade-offs. Examples of such performance data are momentary latency changes, bit rates, temperatures, etc.

The performance data interface is an auxiliary mechanism, as we expect that a production-scale telecom network environment will already have a performance monitoring framework in place.

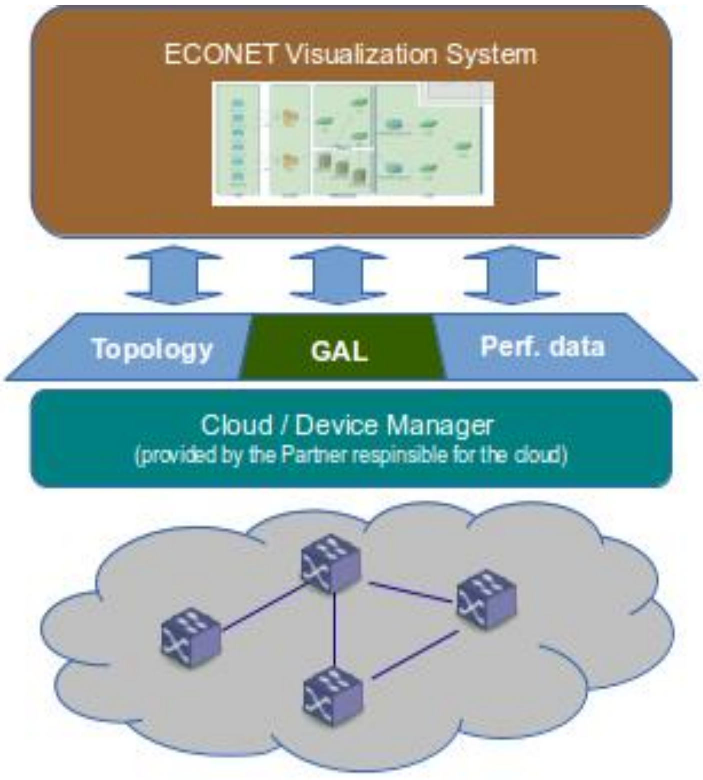

ECONET Visualization System
Topology
GAL
Perf. data
Cloud / Device Manager
(provided by the Partner respinsible for the cloud)

Figure 5. The GAL and other interfaces in the ECONET architecture.

This "protocol vehicle" is a set of XML fragments used over a REST-style HTTP client-server interface.

4.1.2 Introduction to the REST-style interfaces

REST (REpresentational State Transfer), is a popular network API architectural style, which is characterized by the consistent usage of the limited HTTP command vocabulary (GET, PUT, POST, DELETE, etc.), to implement functionality through a series of read and write operations on an object-oriented database. Sections of this database are represented as documents (XML documents in our case) during the client-server communication.

REST is an instance (and competitor) of several similar network API architectural styles, e.g. SOAP, CORBA, DCE, and many other proprietary or language-bound alternatives. It advantages are:

- conceptual clarity,
- fairly easy implementation on both sides,
- it is suitable for proxying, caching and HTTP-level security enforcement.

The basis of REST is a conceptual hierarchical "document" which describes the entirety of the information covered and handled by the API. We call it in our case the "GAL REST XML tree" which describes all information covered by the GAL (including writeable state variables). Since the GAL is hierarchical, the tree is also hierarchical: each level represents a GAL resource, with children corresponding to the subordinate resources (e.g. devices in a network, or ports on a line card, etc.). The "GAL REST XML tree is detailed described in sub-Section 4.1.4.

The GAL REST API works by providing access to parts of this tree through HTTP operations. For example

- discovery is a HTTP GET query where the URL path and the URL query parameters specifying the information to be returned;
- provisioning is a HTTP PUT, with the state specified as an XML fragment in the document part of the HTTP request;
- commit and rollback are POST operations.

In all cases the information returned consists of an error code and/or a document containing the data requested, e.g. for successful operations HTTP returns code 200/OK, while operations on non-existent resources return 404/NOTFOUND, and invalid operations return 401/BAD REQUEST status codes.

Please refer to Section 0. for a detailed description of the operations.

4.1.3 Further network API bindings for the GAL

While this document describes the REST API only, it is possible to define alternative network protocols with the same functionality. Indeed, we have also considered the definition of further “GAL vehicles” like, for example:

- An SNMP API where the resource state is represented as an SNMP MIB, and SNMP protocol (GET/SET/WALK, and maybe TRAP) is used to access this information.
- A CWMP TR-069 binding (called 'profile' in the CWMP world), that provides asynchronous access to the GAL exposed by CPE devices. Please note that since these devices are very simple, the GAL to be exposed is also very simplified. On the other hand, the extended synchronicity of the communication (e.g., delays of up to 24 hours) mandates that the API also exchanges the history of certain states and monitored variables.
- A TMF814-compliant CORBA API to be used with resources that support that protocol.

These alternative bindings are detailed in separate documents.

4.1.4 The GAL REST XML Tree

The principal tag of the XML tree is a GAL resource. Each resource is described by the tag <gal:resource> and it is identified by an ID, usually a variable-length dotted format.

Within the gal:resource tag, the following children nodes supported:

<gal:provisionedState>:	the state provisioned for the resource.
<gal:committedState>:	the state committed for the resource.
<gal:powerStates>:	list of the supported power states of this device.
<gal:monitoring>:	a section of monitored parameters along with their momentary values.
<gal:url>:	this tag indicates that the details of a resource are available from another GAL server (see also Section **Error! Reference source not ound.**4.1.6)

In addition GAL resources can contain further resources described as children:

<gal:physicalChildren>:	physical children.
<gal:logicalChildren>:	logical children.

Please note that this XML tree is not a document, and it does not necessarily exist anywhere in its fullness, but it is rather a conceptual information base, which can be partially queried and manipulated by the commands described below.

```
<gal:resource  gal_id=”0.0.1” description=”AbcAbc1.0”   type=”linecard” >
        <gal:provisionedState state_id=”s0p0”/>
        <gal:committedState state_id=”s0p1”/>

        <gal:powerStates>
                <gal:state state_id=”s0p0”
                                minimum_power_gain=”1.0”
                                maximum_packet_throughput=”10000”
                                maximum_bit_throughput=”12000000”/>
                <gal:state state_id=”s0p1” ….  />
                <gal:state state_id=”s0p2” ….  />
                <gal:state state_id=”s1” ….  />
        </gal:powerStates>

        <gal:physicalChildren count=”n”>
                <gal:resource gal_id=”0.0.1.1” description=”Port 1”   type=”port” >
                        …
                </gal:resource>
                <gal:resource gal_id=”0.0.1.1” description=”Port 2”   type=”port” >
                        …
                </gal:resource>
        </gal:physicalChildren>

        <gal:logicalChildren  count=”n”>
                <gal:resource gal_id=”0.0.1.A” description=”Xyz1.0.1”   type=”logical” >
                        …
                </gal:resource>
                <gal:resource gal_id=”0.0.1.B” description=”Xyz1.0.1”   type=”logical” >
                        …
                </gal:resource>
                ...
        </gal:logicalChildren>

        <gal:monitoring>
                <gal:momentaryConsumption value=”212”/>
                <gal:accumulatedConsumption value=”199” uptime=”38900”/>
        </gal:monitoring>
</gal:resource>
```

4.1.5 GAL operations

In the following all the GAL operations are specifically described.

4.1.5.1 Discovery

HTTP request format

Basic format:

```
        GET   http://<address>/GAL/0.0.1/
```

with an equivalent alternative syntax:

```
        GET   http://<address>/GAL?id=0.0.1
```

and extensible with parameters:

```
        GET   http://<address>/GAL?id=0.0.1&propertySelector=children
```

Query parameters

id =	an alternative way to specify a resource within the GAL.
childDepth =	an integer number indicating the depth of the tree returned, 0 the default returns only the entity addressed.
porpertySelector =	a string specifying the fields of the resource to be returned. Default is **"all"** which cover everything in the REST tree, **except the <monitoring> sections**. Values of this parameter can be any of the following: {**provisionedState, committedState, powerStates, physicalChildren, logicalChildren}**; and the following aggregate selectors can also be used **{all, children, states}**. Multiple propertySelectors can be specified, which effectively selects a union of the specified properties.

Please note that the "monitoring" property section is not addressable by the propertySelector and it is neither included in the "all" aggregate. Consequently, the Monitoring operation (described below in 4.1.5.4 4.1.5.4) has to be used to access it.

Children are enumerated by default values of parameters (childDepth = "0", propertySelector = "all"), but further info on the children is not provided.

HTTP status codes returned

200 OK:	resource found and returned.
404 NOTFOUND:	resource not found.
301 MOVED PERMANENTLY:	the server instructs the client to use a different URI (which is returned). This may happen if the GAL at one level redirects the client to another GAL entity, e.g. a network manager (NMS) GAL redirects a request to the GAL of a certain device.
501 NOT IMPLEMENTED:	this may indicate that certain query parameters are not supported by the GAL serving the request

Document returned

The returned document is a sub-tree of the full GAL tree shown in the previous Section. While the full tree is usually quite extensive, the returned document is normally “clipped” to a manageable size through the childDepth and propertySelector query parameters, e.g.:

```
GET  http://<address>/GAL?id=A.00.0.1&propertySelector=states
<gal:resource gal_id=”A.00.0.1” description=”Card 0.1”
type=”linecard”>
      <gal:provisionedState state_id=”s0p0” />
      <gal:committedState state_id=”s0p1” />
</gal:resource>
```

4.1.5.2 Provisioning

HTTP request format

```
PUT  http://<address>/GAL/0.0.1/provisionedState
```

Query parameters:

None.

Document submitted

A provisionedState section within the REST XML tree:

```
<gal:provisionedState state_id=”s0p2” />
```

HTTP status codes returned

200 OK	
404 NOTFOUND:	resource not found.
501 NOT IMPLEMENTED	

4.1.5.3 Release

HTTP request format

```
DELETE  http://<address>/GAL/0.0.1/provisionedState
```

Query parameters

None.

HTTP status codes returned

200 OK	
404 NOTFOUND:	resource not found.
501 NOT IMPLEMENTED	

4.1.5.4 Monitoring

This operation returns the monitoring sub-tree within the REST tree

HTTP request format

```
GET   http://<address>/GAL/0.0.1/monitoring
```

Query parameters:

None.

HTTP status codes returned

200 OK	
301 NOT_AVAILABLE	
404 NOTFOUND:	resource not found.
501 NOT IMPLEMENTED	

Document returned

It is recommended that the REST server returns 3 metrics for monitoring, as shown in the example below:

```
<gal:monitoring>
      <gal:committedState value=" s0p0" />
      <gal:momentaryConsumption value=" 212" />
      <gal:accumulatedConsumption value=" 199"  uptime=" 38900" />
</gal:monitoring>
```

4.1.5.5 Commit

HTTP request format

```
POST   http://<address>/GAL/0.0.1/committedState
```

Query parameters:

None.

Document submitted

```
<gal:commit/>
```

HTTP status codes returned

200 OK	
404 NOTFOUND:	resource not found.
501 ERROR:	i.e.: nothing to commit.

4.1.5.6 Rollback

HTTP request format

```
POST   http://<address>/GAL/0.0.1/committedState
```

Query parameters:

None

Document submitted

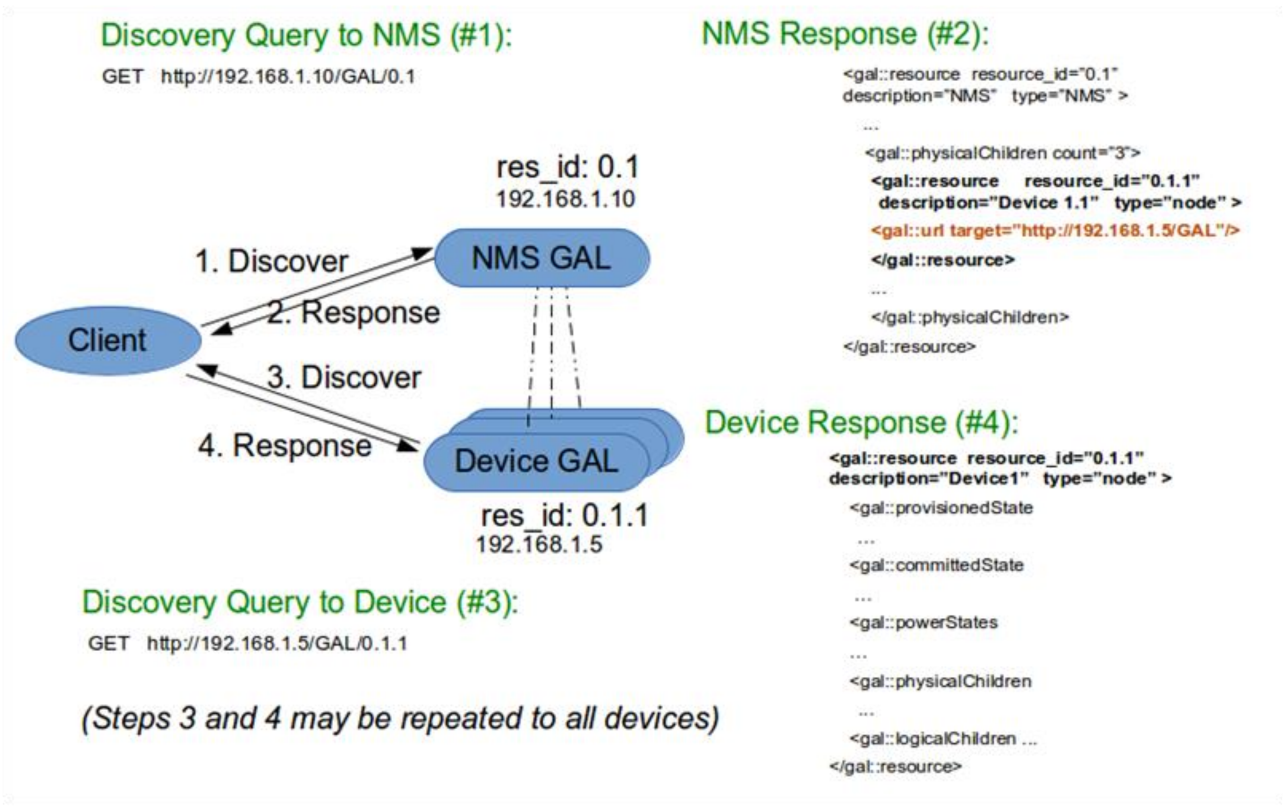
```
<gal:rollback/>
```

HTTP status codes returned

200 OK	
404 NOTFOUND:	resource not found.

4.1.6 Distributed GAL access with REST API - concept

We expect that GAL is not necessarily a centralized information base, but it may also be distributed, in the sense that some information not available (are writeable) at a higher level, may be available at a different level. The most typical example is a network management system, which lists its managed devices, but it cannot tell details of those devices (see Figure 6). This information is accessible by accessing the GAL of the devices themselves only.



```
GET   http://192.168.1.10/GAL/0.1
```

```
<gal::resource  resource_id="0.1"
description="NMS"   type="NMS" >
  ...
  <gal::physicalChildren count="3">
   <gal::resource    resource_id="0.1.1"
    description="Device 1.1"   type="node" >
   <gal::url target="http://192.168.1.5/GAL"/>
   </gal::resource>
   ...
   </gal::physicalChildren>
</gal::resource>
```

```
GET   http://192.168.1.5/GAL/0.1.1
```

```
<gal::resource  resource_id="0.1.1"
description="Device1"   type="node" >
  <gal::provisionedState
   ...
  <gal::committedState
   ...
  <gal::powerStates
  ...
  <gal::physicalChildren
   ...
  <gal::logicalChildren ...
</gal::resource>
```

Figure 6: Example of distributed GAL access: the NMS GAL only contains a reference to device level GALs.

4.2 GAL bindings for Corba / TMF-814 (Section 2)

4.2.1 Architectural overview

The rationale behind the TMF-814 binding is that a significant part of telecommunication equipment EMS and NMS (Equipment Management Systems and Network Management Systems) use exclusively this standard protocol for offering information and control points to external software agents, i.e. to a Network wide vision.

Consequently these systems appear as 'Umbrella Managers', with exclusive authority over the managed devices, and this also requires that the manager reconcile all messages and requests received from partner systems (higher level managers and monitoring systems, like the ECONET dashboard itself, for instance); this complex task primarily influenced the design of the TMF-814 architecture extension to support Power Management.

TMF-814 by itself is built upon CORBA, offering an object-oriented remote API, with extensive functions like naming service, transactions, discovery and notifications, etc. all of which come very useful when thinking about possible applications in the ECONET subject domain. As for Corba objects, native TMF-814 offers several factories, namely managedElementManager, equipmentManager, subnetworkManager, emsManager, protectionManager etc. which provide clean and consequent access to corresponding aspects of the managed device set.

When it comes to ECONET-focused protocol extensions, EMS manager is obviously the primary candidate, with the extension of the lower level equipmentManager and subnetworkManager interfaces, i.e. the ones offering access to module and component-level GAL information. See Figure 7, which reports the extension mapping in the Ericsson device.

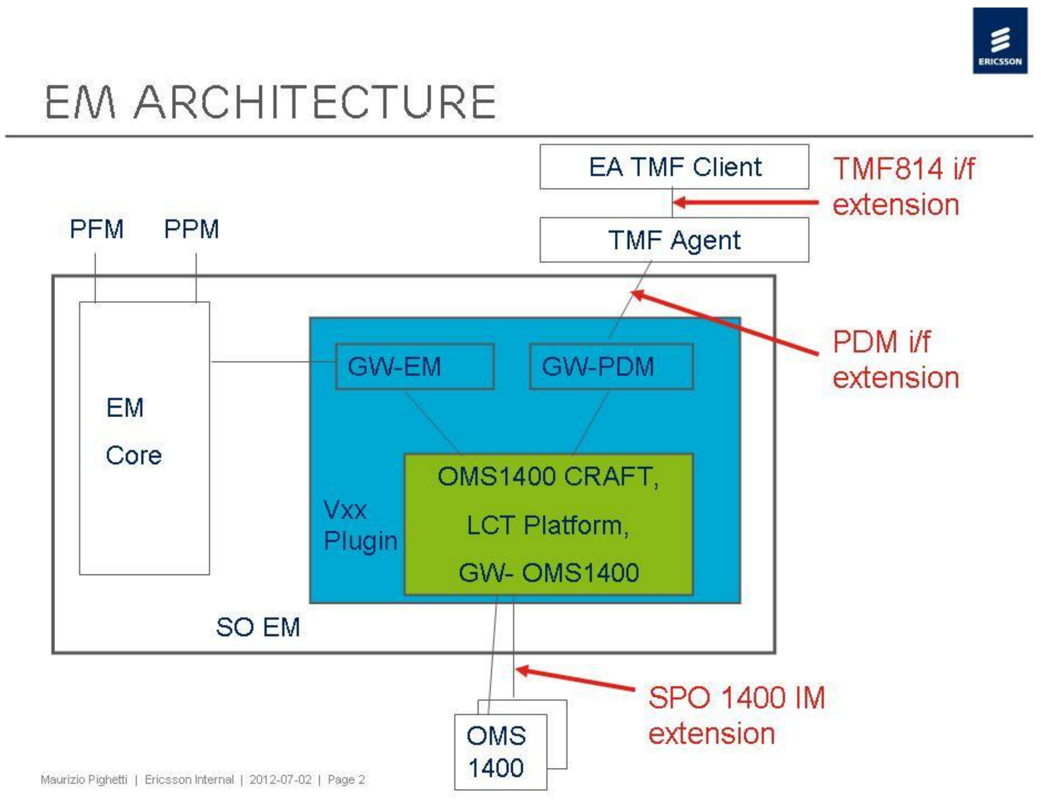


Figure 7. Example of TMF mapping with Ericsson devices.

4.2.2 SPO1400 energy aware

4.2.2.1 EAObject class

Energy Aware SPO1400 prototype has been provided of Energy Aware (EA) Object models, i.e. components of the network element whose energy status can be controlled.

An Energy Aware Object has a aggregation range which goes from 1 to n where n represents the maximum number of ports (on an equipment basis) plus the Voltage, Clock and other type of objects (reset aux, reset board, reset port, reset traffic, etc.). For a complete list see below:

- Voltage
- Clock
- Fan
- Reset_Aux
- Reset_Traffic
- Reset_Board
- Reset_Port
- OptPort_1
- OptPort_2
- OptPort_3
- OptPort_4
- OptPort_5
- OptPort_6
- OptPort_7
- OptPort_8
- OptPort_9
- OptPort_10

4.2.2.2 EAContainer class

An EAContainer object is a collection of EAObjects of different types. An EAContainer is aggregated to a Module.

4.2.2.3 EAEquipmentProfileInfo

The EAEquipmentProfileInfo class models the concept of Power Management Mode (PMM). An EAContainer is associated to an EAEquipmentProfileInfo. An EAEquipmentProfileInfo is a collection of action type assumed by the EAObjects included in the associated EAContainer. Depending on the EA_OBJECT_TYPE, the applicable action types are different. The whole set of action type is described below:

- ON
- OFF
- REDUCED
- INCREASED
- NOT APPLICABLE

Energy capabilities (profiles) are schematically defined in the Annex.

An EAEquipmentProfileInfo is not extensible when the EAObjects action types assume a fixed value in the context of that profile. Its Extensible attribute is set to FALSE.

An EAEquipmentProfileInfo is extensible when the action type associated to some EAObjects can be individually modified (by NMS/LCT).

Only EAObjects of the following types can be modified when inserted in an extensible EAEquipmentProfileInfo.

- OPTICAL_PORT 1
- OPTICAL_PORT 2
- …
- OPTICAL_PORT 10

4.2.3 NBI TMF814 Impacts

North Bound Interface (NBI) TMF814 shall be enhanced to read from EA EMS and report to EA NMS energetic profiles and parameters with actual values. Here following we explain the TMF814 impacts for supporting the SPO1400 energy aware features related to the ECONET project. The strategy is to provide TMF over EM (singleton approach).

Two solutions have been identified:

1) One based on additional information;

2) Another one obtained by adding a new TMF manager that manages the EA networks.

The way to retrieve information from EA EMS is the same in both solutions.

The first solution, definitely suitable for Proof of Concept, is the one developed and implemented in the project, and described in the following. This solution should be applied only as a temporary solution for the Demo because the following:

1) additionalInfo structure to support this kind of information is not the most effective one to model energy information as it results in complex strings to be parsed;

2) in order to read the current energy status of the objects, EquipmentInventoryMgr_I.getEquipment method must send commands to the NE. This behaviour cannot be avoided in case this information is not needed. This introduces delays in this kind of operation.

The second solution (under patent process) is deemed for a future industrially exploitable official release.

4.2.3.1 Energy awareness in TMF

We now describe changes in TMF814 NBI inserted to support the feature.

The following proprietary additional info shall be added to the ManagedElement and Equipment objects, indicating if the ManagedElement or the Equipment is energy aware, based on the field "EnergyAwareSupporting" reported in PDM:

Parameter Name	Applicable MTNM Object Classes or Other Structures	Legal values	AVC notification raised?	Potentially settable from	Comment / Example
“EnergyAware”	Equipment, ME	“True”, “False”	no	-	-

If the field "EnergyAwareSupporting" is not present in PDM, then no additionalInfo shall be added. In this case the meaning is that the NE and/or the card are not energy aware.

4.2.3.2 Energy Aware Management

In this solution we add proprietary TMF manager and specific methods and objects to manage EA resources (see Figure 8 for the basic structure). Also the way to report the information to the EA NMS is proprietary. As specifically tailored for TMF814, this simplistic approach cannot be extended to Multi-Technology Operations System Interface (MTOSI). Additional developments, for the second solution previously cited, are foreseen in order to create the MTOSI related structures and manager.

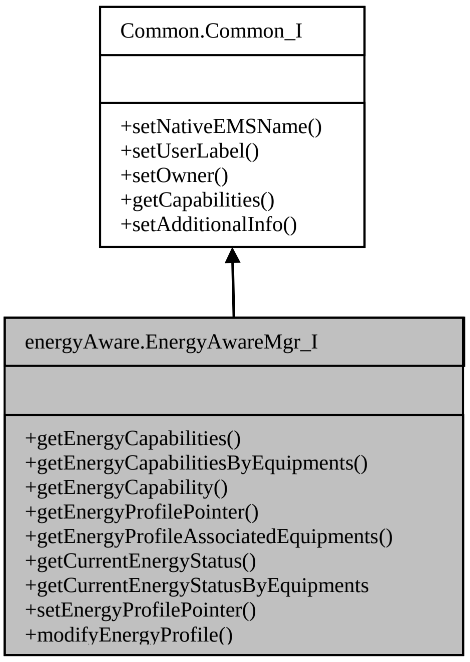

Common.Common_I
+setNativeEMSName()
+setUserLabel()
+setOwner()
+getCapabilities()
+setAdditionalInfo()
energyAware.EnergyAwareMgr_I
+getEnergyCapabilities()
+getEnergyCapabilitiesByEquipments()
+getEnergyCapability()
+getEnergyProfilePointer()
+getEnergyProfileAssociatedEquipments()
+getCurrentEnergyStatus()
+getCurrentEnergyStatusByEquipments
+setEnergyProfilePointer()
+modifyEnergyProfile()

Figure 8. Synthesis of the energyAware.EnergyAwareMgr_I structure.

4.2.3.3 Energy capabilities realignment

TMF shall be able to realign the energy capabilities (profiles) calling the PDM message “GetEnergyCapabilities”.

After the realignment of the physical configuration, if the property “EnergyInformationManagement” in i36PlugIn.properties files is present and set to true, for those ManagedElements that have “EnergyAware” additionalInfo set to “True”, TMF application shall send the PDM message “GetEnergyCapabilities” and shall read the energy capabilities.

The way to report the information to the EA NMS is based on the additionalInfo structure and on the methods to set and get it: Common_I.setAdditionalInfo, ManagedElementMgr_I. getManagedElement, EquipmentInventoryMgr_I.getEquipment, etc..

4.2.3.4 Realign Current Energy status

At TMF start-up, after the realignment of the configuration, if the property "EnergyInformationManagement" in the i36PlugIn.properties file has the value true, the current energy status shall be retrieved calling the PDM method GetCurrentEnergyStatus described in Section 4.2.3.5. The method shall be called at NE level, for each NE that has the additionalInfo "EnergyAware" set to "True".

This message shall report information about the profile applied on each card, other than information related to the current energy status. Specific additionalInfo shall be added to the related Equipment. In case the profile is "extensible", additionalInfo shall be added also to the related ports.

As per the requirements, EA NMS shall ask for the current energy status per card. The VXX plug-in reports energy status for each component of the card; then, the NBI TMF shall process the information to report to EA NMS a view per card.

4.2.3.5 Get Current Energy status

EA NMS can retrieve the current energy status, by calling the TMF method EquipmentInventoryMgr_I.getEquipment on the equipment.

This shall cause the call to PDM method GetCurrentEnergyStatus previously described, but in this case at the card (Module) level. This is subject to the property "EnergyInformationManagement" in i36PlugIn.properties: the PDM method shall be called if this property has value true.

As for the realignment, TMF shall export current energy information to EA NMS per card. The current energy status shall be updated and stored for each card using the additionalInfo (see Section 4.2.3.4).

4.2.3.6 Get Current Traffic Load

The EM-VXX plug-in will report the attributes covering performance data to retrieve Actual Traffic Load through a specific request via PDM (GetCurrentStatus).

Actual Traffic Load, i.e. CurrentDataRateRx/Tx of all EA Objects belonging to the card, can be retrieved by EA NMS by means of polling the relevant information (packets counter attributes, reported in CurrentDataRateTx/Rx information), correlated to a time window. TMF shall also make available the following data in the additionalInformation field:

1) BWTotalReduction;
2) MinBWThreshold;
3) MaxBWThreshold.

Correlating BWTotalReduction with CurrentDataRate information it is possible to retrieve the Actual Traffic LoadBWTotalReduction as the bandwidth reduction at the given PMM compared to maximum bandwidth, i.e. the Actual Max Traffic Load expressed in % of the maximum bandwidth of the given EA Container according to the applied PMM.

For example, if the EAContainer is at maximum bandwidth, BWTotalReduction shall be 0%, if a PMM is applied to the EAContainer, BWTotal Reduction shall be x%.

CurrentDataRate is the Actual Traffic Load of the given EA_Container, reported in bps. By means of this value, correlated with Actual Max Traffic Load expressed in bps of the given EA Container according to the applied PMM (the Actual Max Traffic Load in bps can be retrieved by the BWTotalReduction % and by the maximum bandwidth in bps of the given EA_Container), the EA NMS shall understand the level of utilization of the given EA Container according to the applied PMM.

By calling the TMF methods EnergyAwareMgr_I.getCurrentEnergyStatus() and EnergyAwareMgr_I.getCurrentEnergyStatusByEquipments(), which will report, among other information, the CurrentDataRate of the related EAContainer, this information can be retrieved.

MinBWThreshold, MaxBWThreshold shall report the information related to the minimum bandwidth under which the component is under-utilized and the maximum bandwidth over which the component is near to be over-utilized, respectively .

Notifications shall be sent when the actual traffic load is higher/lower than the configured threshold values.

4.2.3.7 Set Power Management Mode

It shall be possible for the EA NMS to change the energy profiles on the cards and, for those profiles that are extensible, to change the status of some objects.

The way to change and modify energy profiles is by using the Common_I.setAdditionalInfo method, calling it on the Equipment.

1) Call PDM “ProfileName” field. It is the profile that shall be applied; it shall be set with the value of “EnergyProfileName” additionalInfo passed in the setAdditionalInfo method
2) The PDM “ProfileId” field shall be set with the progressive number of the “EnergyProfile_<n>” additionalInfo that corresponds to that “ProfileName”. This has to be retrieved from “EnergyProfile_<n> additionalInfo stored on the ManagedElement
3) Optional PDM field “EAObject” shall be set based on the “name” of “OptPort_x” additionalInfo of the Equipment
4) Optional PDM field “PortAction” of the message shall be set based on the new “value” of “OptPort_x” present in setAdditionalInfo message. All the possible values are listed here; the only values to be used by EA NMS are “ON” and “OFF”.

If the response is ok, then the “EnergyProfileName”, all the “OptPort_x” and other EA Objects additionalInfo shall be updated accordingly, otherwise the value shall be unchanged. Setting of “EnergyProfileName” and “OptPort_x” additionalInfo through a single setAdditionalInfo() call, shall cause the creation of a single PDM message. This will be the preferred way to set PMM.

4.3 GAL bindings for TR-069

4.3.1 Introduction

The section describes a proposed mechanism for controlling and monitoring the energy behaviour of CPE devices, in a massive, “carrier-grade” scale and robustness. The mechanism is built upon the Broadband Forum TR-069/CWMP specification, which is the preferred way for simultaneous remote configuration and monitoring of a large number of customer devices.

Another design goal was the alignment of this communication framework with the principles and concept of the Green Abstraction Layer described in the document D4.2 of the ECONET consortium. Specifically, the proposed mechanism addresses the following capabilities:

- Hierarchical decomposition of the device along the “tree” of power aware and/or individually identifiable power consumer components (noting that for typical CPE equipment the 'hierarchy' is typically small and shallow).
- Capability for controlling power modes administratively (e.g. by a higher level manager), or letting the device control power modes autonomously.
- Accessing a history of power modes and metered values through an intermittent and relatively infrequently available communication link.

4.3.2 Considerations

Configuring and monitoring of the actual values in CPE devices is a pretty straightforward and proven technique in TR-069, e.g. it is in mainstream use by operators for configuring customer-specific home gateway settings like VoIP numbers, firewall rules, etc.. The corresponding relevant Recommendation also suggests the use of TR-069 for historical type of status information (i.e. performance monitoring), but the way to do this is not detailed, and consequently such mechanisms are not provided by typical TR-069 devices and auto-configuration servers.

The problem here is that we want to collect information (power modes and metered values) characterized by the following properties:

- The data is often constant or changes very slowly and in a relatively narrow range.
- Power mode changes, however, result in sharp changes, and a sequence of such changes may occur in a short period.
- The information is used primarily for large-scale analysis of CPE power control mechanisms and the resulting behaviour; thus, a constant and/or frequent data link between the manager and the CPE devices is not justified. Rather, it is preferred to stick with the typical CPE WAN Management Protocol (CWMP) provisioning cycle times like 2 to 24 hours.

4.3.3 Solution architecture

The proposed power control and monitoring scheme is illustrated by Figure 9 below. The main design goal here was clearly to leverage server components, which are often available as part of the Operational Support System (OSS) stacks of operators. Thus the server side consists of:

- A TR-069 Autoconfiguration Server (ACS), with „northbound” interfaces and capabilities of accepting commands for “discovery”, “assignment” and “query” on CPE devices.

- A performance monitoring server prepared to collect information through the ACS interface, and to store, analyse and display this data in human readable format and also as data to be processed by other applications.
- A Network Policy Controller which implements the power control mechanisms either as a separate "power policy controller" or as part of a more general central inventory and provisioning system for CPE devices, communication links and other features.

On the Customer Premises side, the figure not only displays the customer home-gateways, but also additional CPE devices (TVs, phones, computers and also other electrical devices), which are possibly controlled through the gateway. While the "home network" is out of scope for the ECONET project and consequently this is not described in detail, extending the "green control plane" to home devices offers huge potential power savings, and is a viable option, well worth to be explored.

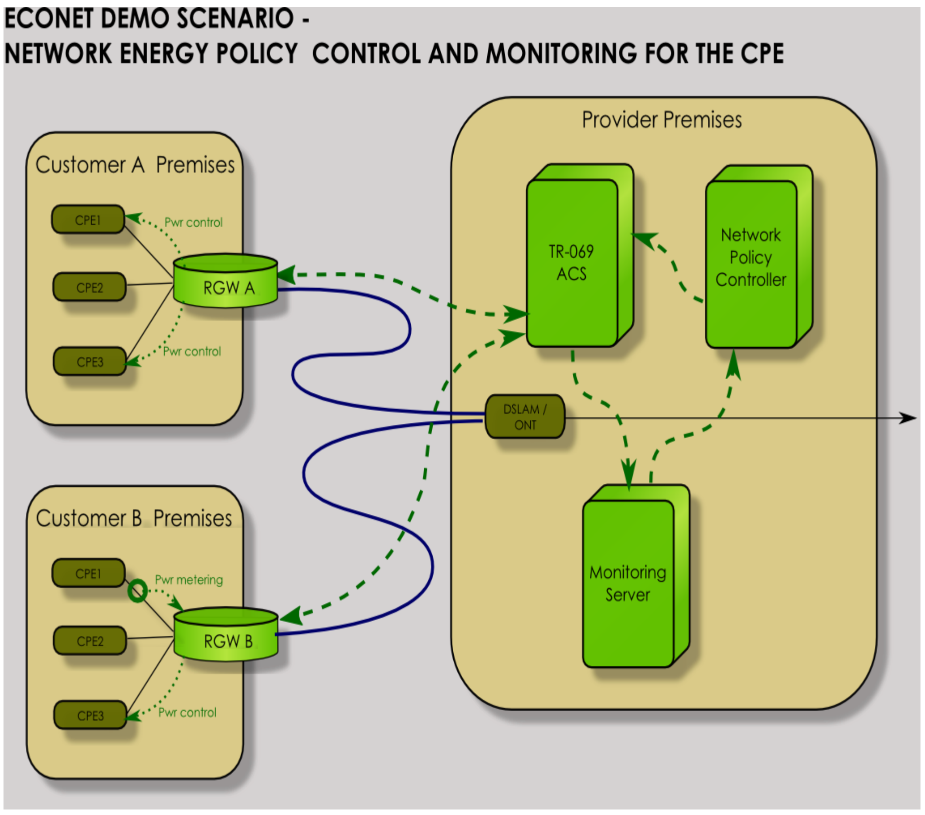


Figure 9. Power control and monitoring scheme

4.3.4 Description of the Power Data Model

A key interface of the architecture is the one between the ACS and the managed devices. This interface is used over the TR-069 protocol, i.e. it is a TR-069 data model (see Annex I).

The model defines methods for controlling the power economy behaviour of devices, and also for monitoring their power modes and power usage. The model allows for implementing these features in several compliance levels, i.e. profiles:

- The Baseline profile requires the support for putting the device into one of several power states, and also for reading the momentary value of electric power and accumulated energy usage. These values may either be calculated or measured.
- The Component profile provides capabilities for the inspection and control of power behaviour of some parts of the device.
- The History profile extends this by the capability of getting historical data for power modes and usage. This profile makes it possible to implement CPE power monitoring in large, provider-wide scale CWMP-ACS systems, where the frequency of data exchange with the individual CPE-s is rather low, (i.e. several hours or days).

4.3.5 Collection of History items

The semantics of the History is described below in more detail. The goal here is to provide a mechanism to accommodate the acquisition of both long-term slow-changing and abrupt, fast changing information (as described in the Section 4.3.2 above).

The mechanism is a hybrid of periodic and event based data collection as showed in Figure 10 below. There is a periodic, low frequency data collection with additional collection points inserted when the change of some parameters exceeds certain thresholds. Considering the power domain there are two types of thresholds suggested:

- Any change of the actual power state of a device or a component within (i.e. a change of `PowerState` or `PerformanceMode` below)
- A threshold of 10-20% is recommended for changes on the usage values, i.e. on the `MomentaryWattage` values below.

The figure below illustrates the collection periods resulting in a scenario that includes examples for both threshold crossings.

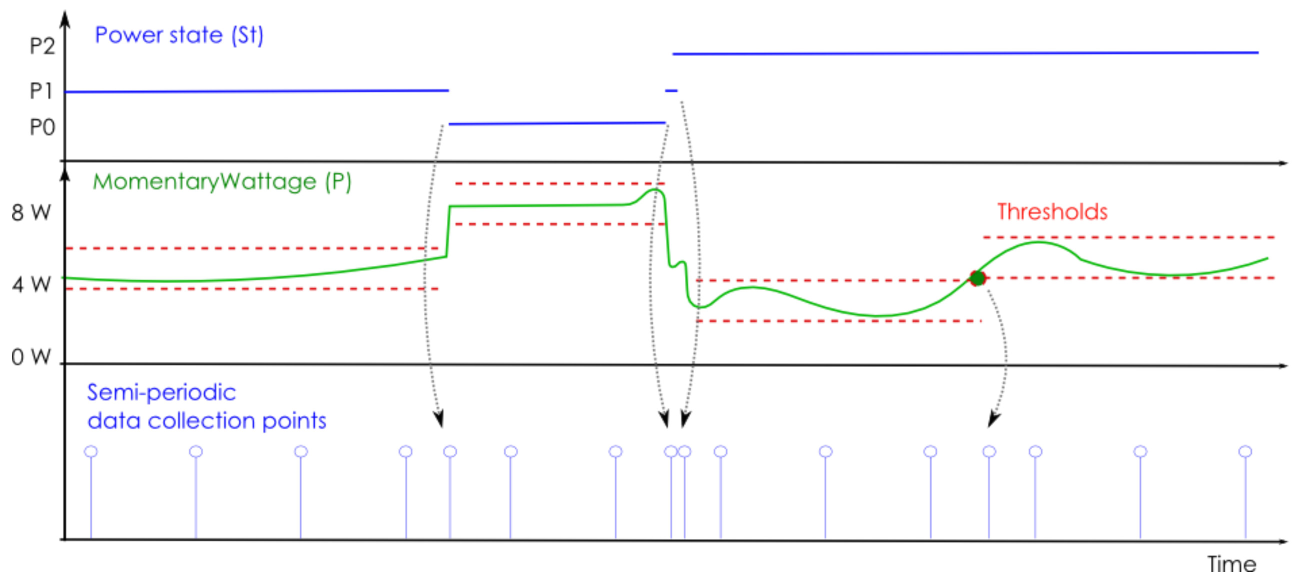


Figure 10. Semi-periodic history data collection.

5 Green extension of OPENFLOW

5.1 Introduction

The evolution of network and networked device architectures are two of the most important aspects of the Internet as we know it today, since they directly reflect the main open issues and the needs of current and upcoming network technologies and services.

The first aspect that is driving the evolution of the Future Internet is certainly the global increase of network traffic: in **Error! Reference source not found.**, Cisco estimated that global IP traffic as increased eightfold over the past 5 years, and will increase fourfold over the next 5 years. Moreover, Internet video was estimated to account for over 50% of consumer Internet traffic by 2012. Since video traffic has strict Quality of Service (QoS) requirements, the service level must be sufficient to support such services.

The estimated increase of IP traffic has also drawn attention to the power consumed by the ICT. The Global e-Sustainability Initiative (GeSI) **Error! Reference source not found.** estimated an verall network energy requirement of about 21.4 TWh in 2010 for European Telcos, with a figure of 35.8 TWh in 2020 if no Green Network Technologies (GNTs) will be adopted. Considering that energy costs are raising quickly all over the world, the IP traffic increase trend would soon be unbearable if current network architectures and technologies were maintained.

Hence, re-thinking and re-designing the Internet becomes inevitable to keep the pace of any technological improvements. For this reason, many current and upcoming research efforts are emphasizing the weaknesses of the Internet infrastructure identifying the main issues and proposing novel solutions to guarantee a viable evolution of the Future Internet.

The most common criticisms regard the lack of flexibility of the currently deployed solutions: in fact, network infrastructures have not changed much throughout the years, being still based on the TCP/IP paradigm, and hence not providing an efficient degree of integration between network architectures and services. As a result, this architectural ossification has in many cases delayed the introduction of new technologies. In order to support novel solutions for the Future Internet development, programmability will play a key role. Moreover, although energy efficiency aspects are gaining more and more interest among both researchers and manufacturers, their actual introduction inside networks and network devices is still far from becoming a custom. Energy efficiency does not represent the mere reduction of CO_2 emissions, but the achievement of a satisfying trade-off between power consumption and network performance. Unfortunately, it is very difficult to save energy without affecting the correct execution of network services.

In order to address the increasing need for flexibility of the Future Internet, Software Defined Networking (SDN) has recently emerged as a ground-breaking technology for realizing flexible network nodes able to customize their behaviour according to the network and service needs. Although it represents a recent solution, SDN is rapidly evolving and gaining attention from both researchers and manufacturers. As stated in **Error! Reference source not found.**, SDN will be a 3.52 billion global market by 2018 thanks to the growth of cloud services and the need for mobility.

Among all existing realizations of the SDN paradigm, OpenFlow (OF) **Error! Reference source ot found.** is probably the most popular one, reaching a point where it is often considered a synonym for SDN. Originally conceived to allow the testing of new protocols inside campus networks through the separation of the traffic flows, OF is characterized by the separation of the data- and control-plane. In detail, the data-plane still resides in the network node (called OF switch) and performs flow-based traffic processing: packets are matched against the entries of a flow table and are then forwarded or manipulated (modify L3/L4 header fields, push/pop a VLAN tag, …) according to the corresponding flow table entry. The central Controller is in charge of making decisions on unmatched packets and in general performing control-plane operations. An example of an OF network can be seen in Figure 11.

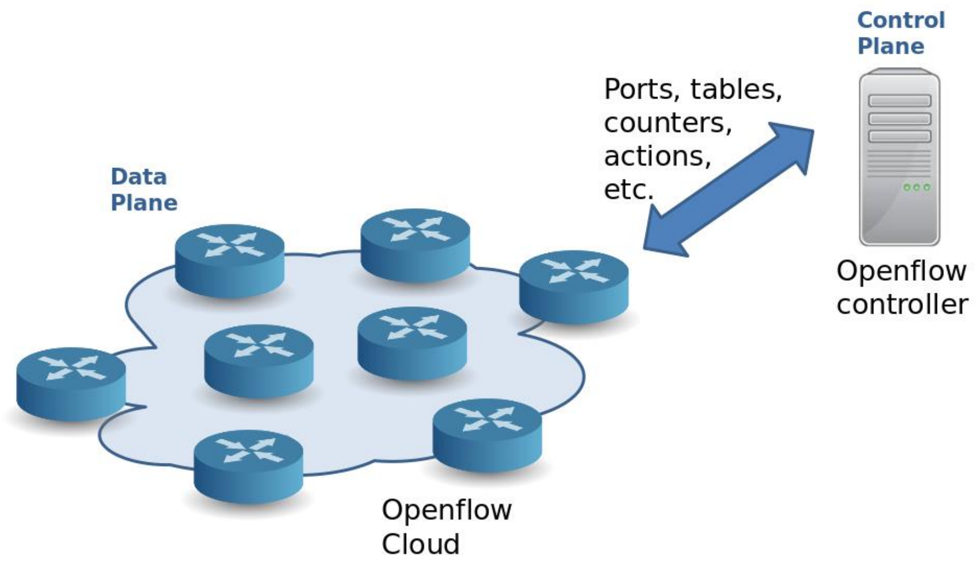


Figure 11. OpenFlow most frequent deployment.

Hence, the Controller plays a very important role in an OF network. It guarantees communication among hosts using a standard language called OF protocol, which implements a set of primitives to manage the flow table of each node in the network and to obtain a complete knowledge of the node itself. The centralized architecture typical of OF represents a valid solution for guaranteeing the correct interaction of all network elements. Another strength is represented by the standard communication system provided by the OF protocol and the way it collects and distributes information about the traffic demands and the nodes status.

For these reasons, we think that this architecture, which presents a hierarchical structure and a standard communication channel, would be the perfect support for introducing energy-aware network control policies. Such "green" enhancements would allow the allocation of traffic flows and the managing of the internal states of the single nodes at the same time: with this strategy, network consumption would be minimized and the correct bandwidth allocation for each service would be guaranteed.

To the best of our knowledge, there is no technique which makes the controller aware of the power consumption of the internal hardware components of the switches: we candidate the Green Abstraction Layer (GAL), defined in WP4 (see [1-6]), to this purpose. The GAL represents an internal communication interface/middleware to exchange power management data among packet processing elements, and apply control strategies in a simplified way. In this respect, the purpose here is to integrate the GAL primitives inside the OF protocol to manage the network power configuration along with the traffic flows: the expected result is the dynamic and optimal configuration of the path between a source and a destination from the energy point of view. In order to show a quantitative evaluation of the gain achieved with this integration, we propose in the following an algorithm to solve the in-network consolidation problem.

5.2 Network Processor Architecture

Next-generation network processors need to move forward from the typical proprietary architectures to being more flexible and support the capabilities needed for developing a Future Internet network. As mentioned in the Introduction, such processors must be easily programmable and customizable, which is usually obtained through the adoption of a central intelligence and a number of function-specific HW components with some level of “energy-awareness”.

Today, one of the most popular trends consists in using multi-core architectures for network processors. The main reason is that multi-cores allow increasing network performance without using too high clock frequencies, which typically cause heat dissipation and data synchronization issues. In this respect, most solutions are derived from PC-based SW routers or Systems on a Chip (SoC) platforms. The former have represented a diffused trend in the research for a long time. Their advantages are ease of programmability (especially in presence of the Linux Operating System) and low costs. In addition, PC platforms already embed power management capabilities, usually controlled through the ACPI standard. The main criticism that has been moved to SRs in the past concerns their performance level, which has not been comparable to commercial routers for a long time. However, the introduction of specific HW able to improve the system efficiency, such as NICs that can guarantee some packet pre-processing, has boosted their potentials. SOC platforms represent a better performing and, of course, more expensive solution for the design of programmable network processors. They are characterized by a number of dedicated HW components to perform various kinds of operations outside the central cores, and also include various forms of power management capabilities.

Network processors, either PC- or SoC-based, present very similar architectures when it comes to the packet elaboration. Even proprietary architectures work in the same way, but of course they do not provide so many freedom degrees as programmable/configurable devices. A simple example of this architecture can be seen in Figure 12.

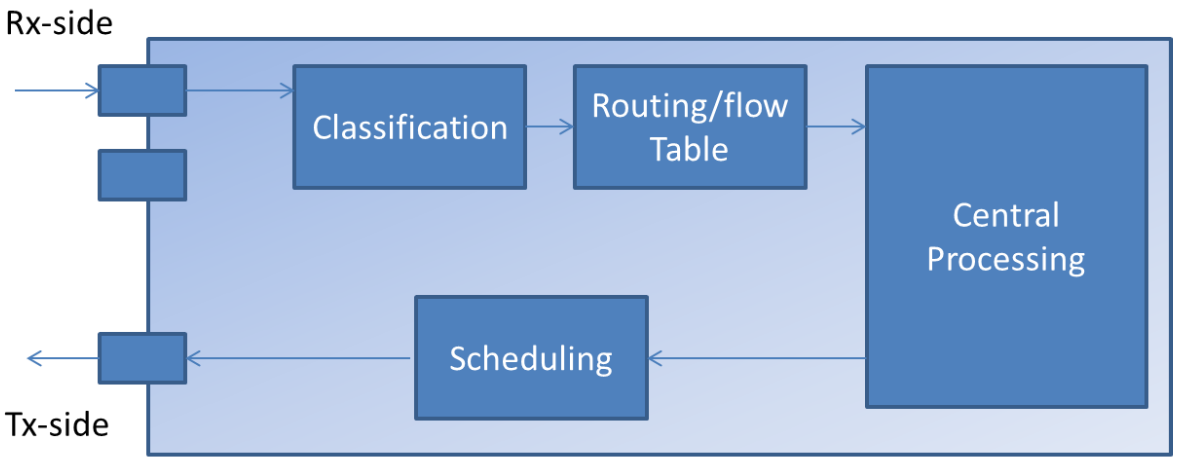


Figure 12. Network processor architecture.

Packets are received by one or more NICs, which might provide some pre-processing operations, like packet classification and table, or such operations might be performed inside the system but in a similar manner.

In detail, packet classification is usually performed by a HW parser, which extracts pre-selected header fields from the incoming packets and matches the so composed key against the entries of a flow table. The efficiency of this building block is crucial to guarantee high-performing packet processing. For this reason, the memories used for these tables have become more and more complex to provide fast matching capabilities.

RAM memories, which represent the most diffused support for flow tables, are known to take a memory location as input and return the value contained at that address. In contrast, Content Addressable Memories (CAM) work exactly in the opposite way, as they perform an exact match of data and return the address at which they are contained. This characteristic makes CAM a very interesting technology for building flow tables. However, since only exact matches can be performed on a CAM, it is really suitable only for flow matching, but not for prefix matching. A possible solution could be that of "expanding" each prefix into all possible flows but that would quickly exhaust the memory space.

A more flexible type of CAM is represented by Ternary CAM (TCAM). Ternary means that, in addition to the binary "0" and "1" digits, there is also an "X" symbol representing a wildcard choice. The inclusion of wildcards allows non-exact matching, hence the possibility of prefix matching. The main drawback of TCAMs is represented by their high power consumptions: in fact, this parallel structure takes a lot of energy to power the match line, while wasting most of the energy on non-matches. The possibility of decomposing a single TCAM into logical sub-tables would allow independently managing their power configurations, for example by turning off the power supplied to the empty memory portions.

In Linux-based platforms, forwarding is usually performed at kernel level. Although this strategy allows covering all functionalities needed in a forwarding engine, and the parallelization of the forwarding process among cores allows strongly improving performance, some recent solutions based on moving forwarding from kernel level to user-space level have provided massive performance improvements, at the price of a reduced number of network functionalities. Since the framework runs in user-space, the high overhead associated with kernel operations and with copying data between kernel and user space is removed. Moreover, significant time is also saved with further improvements, like disabling interrupts generated by packet I/O. In this respect, one of the most promising solutions is represented by the Intel Data Plane Development Kit (DPDK).

At the Tx side, simpler systems do not perform any further operations. If supported, a scheduling discipline allows processing traffic flows respecting their priority level, which means a higher bandwidth percentage is assigned to traffic with stricter QoS requirements.

The architecture of a network processor, with the "cohabitation" of separated Rx and Tx processing chains, maps perfectly with the OF protocol operations. Moreover, the logical distinction between data- and control-plane typical of such devices clearly corresponds to the centralized structure of OF. Finally, the way such centralization can be exploited for a more hierarchical purpose will be clarified in the next section.

5.3 The OpenFlow Protocol Extension towards Green Capabilities

OpenFlow specifies an interface, called OF Channel, used to connect each switch to a controller. The implementation of the channel can be vendor specific as long as the messages are compliant with the OF Protocol. Such messages can travel from the Controller to the network node, when information is required, or vice versa. In this respect, our aim would be to exploit the OF Channel characteristics to integrate its set of messages with some primitives concerning the power state of the network devices inside the network, like available power configurations and relative behaviour (e.g., power consumed by the device in that particular state), or current power state. As mentioned in the Introduction, these communication primitives are already specified by the Green Abstraction Layer.

The GAL allows exchanging power management data and applying control strategies in a simplified way. Its application is driven by two main capabilities. On one hand, it hides the implementation details of energy-saving approaches. On the other hand, it offers a standard interface to guarantee interactions between heterogeneous green capabilities and HW technologies, as well as between control and monitoring frameworks. The power management settings are driven by the Local Control Policies (LCPs), control processes developed to optimize the router configuration in order to achieve the desired trade-off level between energy consumption and network performance according to the incoming traffic load. The GAL has a hierarchical structure, where multiple instances are used for managing the exchange of information between the LCPs of different levels and between LCPs and hardware components.

By means of the GAL, control applications are allowed to get information on how many power management settings are available at the data-plane, and on the potential effect of using such settings. The other way around, control applications are capable of setting a certain power management configuration to the device data-plane. It is worth noting that, in order to provide the information needed by control applications to reduce the energy consumption while meeting QoS constraints, the GAL must explicitly represent the impact on network performance when different power management settings are applied.

An OF switch can be represented, from a network point of view, by a set of ports (Physical, Logical and Reserved), a set of actions that can be performed on traffic flows (the list of actions of a switch depends on the specific hardware) and one or more flow tables. An OpenFlow version of the GAL can consider as logical entities the ports, the actions and the tables of all the OF switches. Since there is no constraint about how those resources have to be implemented inside the device, manufacturers can use their own technologies and solutions and thanks to the GAL each resource can be defined in terms of power consumption with respect to the offered load. In this way, all the entities of the network can be completely described, from the energy perspective, by a set of EAS.

The extension of the OpenFlow protocol to make the network efficient using the GAL requires both the definition of new OpenFlow messages and the design of a new module inside the latter.

As far as the new message definition is concerned, the network nodes have to send the description of their own logical resources and related EAS to the controller. In this direction, we have defined a set of new messages to implement the main GAL primitives: discovery, provisioning and monitoring.

The discovery primitive allows the Controller to get information about the power consumption of each entity. The related packet from the network node to the Controller contains the complete description of the EAS: an identifier, the power gain with respect to the maximum power consumption and the throughput in terms of packets per second and bits per second. In order to completely support the GAL, the message also contains wake-up time, sleep time and transition power.

The provisioning primitive allows the Controller to set the state of a network node. The related packet from the Controller to the network node contains the logical resource identifier and the EAS identifier.

The monitor state and monitor consumption primitives allow the Controller to get information respectively about the current EAS set on a logical entity and the current power consumption of the entity. The related packets from the switch to the Controller contain the identifier of the entity and, in the first case, the identifier of the current state, in the second case the power consumption of the entity.

In addition, we propose a new Controller module called Energy Optimizer: it is a possible demonstration of how the controller can exploit the information provided by the GAL to reduce the power consumption of the network. In fact, the Controller gathers information about the EASs, and the Energy Optimizer module calculates a nearly optimal configuration to process a given traffic demand consuming the minimum energy.

5.4 In-Network Consolidation

In the previous sections the advantages that can be obtained through the introduction of power management capabilities inside the OF protocol, in terms of energy efficiency and resources management, have already been roughly sketched. Now we want to move a step further and provide an example to clarify how our study would help overcoming scalability issues and promote energy efficiency at the same time. To do this, we can start by describing the network management as a consolidation problem.

The term consolidation describes the optimal allocation of functionalities/resources to guarantee a certain outcome. The concept of consolidation usually refers to the Virtual Machines (VMs) allocation in a variable number of physical servers inside a datacenter. Since we aim at extending the same concept to the optimal allocation of network resources, in the following the tackled problem will be called in-network consolidation.

5.5 The In-Network Consolidation Algorithm

This section presents an algorithm to calculate the solution of the "in-network consolidation" problem. We consider a network, as represented in Figure , composed by R routers connected by L links. Each router, aside from forwarding, can also perform A possible off-loading operations on traffic, or can be turned off when it is not required, like the bottom left one in the figure.

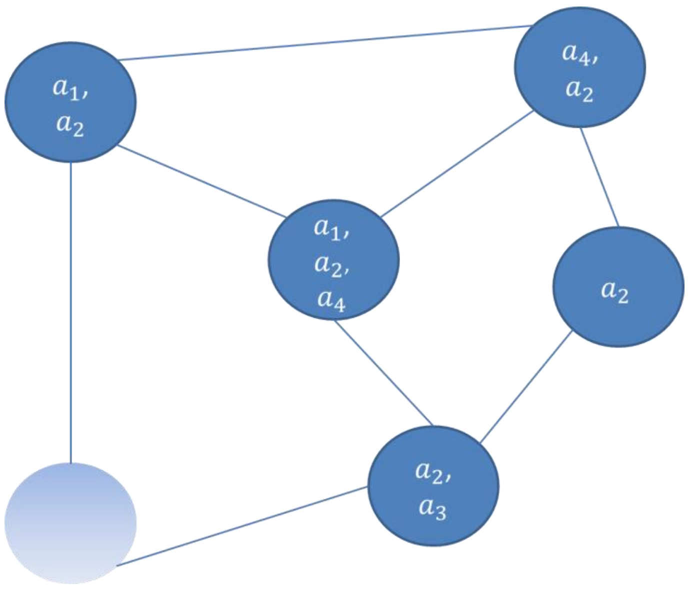


Figure 13. Representation of a network with sleeping capabilities.

The traffic demand is represented by a set of F flows, with the generic f flow defined by:

- a source s_f and a destination d_f
- a load L_f
- a subset of of the available A actions required by the flow.

Here, we formally define a problem, which takes as input the topology, the logical resource and the traffic demands, and finds the flows and actions configuration that satisfies the traffic demand and minimizes the power consumption of the entire network. Table.1 contains the description of the variables and parameters that are used for the problem formulation.

Table 1. Notation definition.

u_{lf}	binary variable that equals 1 if the link l is crossed by the flow f
v_{rfa}	binary variable that equals 1 if the router r crossed by the flow f implement the action a
x_l	binary variable that equals 1 if the link l is active
y_{ra}	binary variable that equals 1 if router r enables the action a
z_r	binary variable that equals 1 if the router r is active
R	total number of available routers
L	total number of available links
A	total number of available actions
F	total number of flows
T_r	activation cost of the router r
U_l	activation cost of the link l
V_{ra}	activation cost of the action a on the router r
L_f	load of the flow f
M_l	capacity of the link l
w_f	binary constant that equals 1 if the flow f requires the action a
h_{rl}	matrix of binary constants that equal 1 if the link l enters the router r
k_{rl}	matrix of binary constants that equal 1 if the link l exits the router r

$$\min_{u,v,x,y,z} \sum_{r=1}^{R} \sum_{l=1}^{L} \sum_{a=1}^{A} T_r + U_l x_l + V_{ra} y_{ra} \qquad (1)$$

$$y_{ra} \geq \frac{1}{F} \sum_{f=1}^{F} v_{rfa} \qquad (2)$$

$$\sum_{r=1}^{R} v_{rfa} = w_{fa} \qquad (3)$$

$$\sum_{f=1}^{F} L_f u_{lf} \leq M_l \qquad (4)$$

$$\sum_{l=1}^{L} h_{lr} u_{lf} - \sum_{l=1}^{L} k_{lr} u_{lf} = 0, r \neq s_f, d_f \qquad (5)$$

$$\sum_{l=1}^{L} h_{lr} u_{lf} - \sum_{l=1}^{L} k_{lr} u_{lf} = 1, r = s_f \qquad (6)$$

$$\sum_{l=1}^{L} h_{lr} u_{lf} - \sum_{l=1}^{L} k_{lr} u_{lf} = -1, r = d_f \qquad (7)$$

$$x_l \leq z_r \qquad (8)$$

$$v_{rfa} \leq \sum_{l=1}^{L}(h_{lr} + k_{lr)}\, u_{lf} \qquad (9)$$

$$x_l \geq \frac{1}{F}\sum_{f=1}^{F} u_{lf} \qquad (10)$$

$$u_{lf}, v_{rfa}, x_l, y_{ra}, z_r \;\in\; [0,1] \qquad (11)$$

Equation 1 is the cost of the whole network and is composed by three components, representing the presence of each active router, port and action. Equation 2 states that the action a has been enabled on the router r for all the flows requiring it, while Equation 3 guarantees that, for a given flow f, each one of its actions are performed only ones, and Equation 4 that the capacity of each link is respected. Equations 5 - 7 represent the flow conservation constraints. Equations 8-10 determine the number of routers and links that are used for forwarding. Finally, Equation 11 constraints the values that can be assumed by the problem variables.

5.6 Test Results

The problem presented in the previous section has been applied to the simple topology shown in Figure . This simple network is composed of five routers connected by twelve mono-directional links. Four actions are made available on the network and can be activated according to the demands. In this first case, the activation cost is 5 for each link and for each action, and equals to 20 for each router. The capacity of each link corresponds to 1.

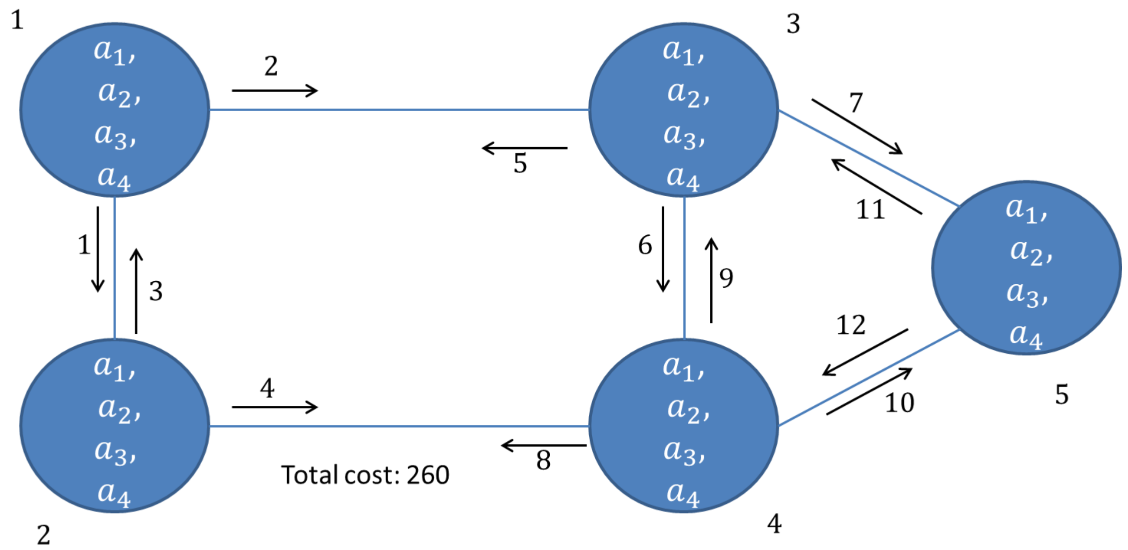


Figure 14. Topology of the network used for testing the optimization problem.

The traffic demand that has been chosen for this example consists of three flows, which have the characteristics reported in Table 2.

Table 2. Traffic Flows

Flow	Load	Source	Dest	Actions
f_1	0.1	1	5	1, 3, 4
f_2	0.1	2	3	2, 3
f_3	0.1	1	3	2, 3, 4

Figure shows the flows and actions allocation obtained through the algorithm presented in the previous section. The optimization strategy allows managing all flows using only 3 of the available 12 links, and also switching off Router 4. Moreover, each action is enabled only once on the network and serves all flows requiring it. As a result, the total cost of the network is around 56% lower in the optimized configuration.

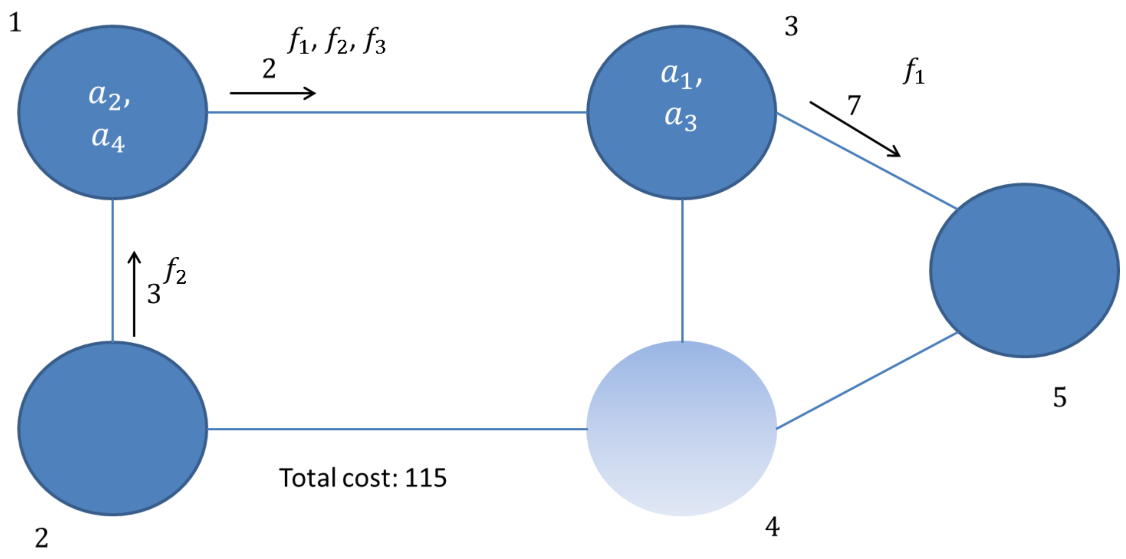


Figure 15. Optimized flows and actions allocation.

In the next example, presented in Figure , we suppose to increase the load of f3 from 0.1 to 1. This new parameter brings to a stress situation in the network configuration: in fact, the load of f3 now corresponds to the link capacity, hence no other flow can be transmitted along with f3. For this reason, the flows allocation requires the activation of the sleeping router and of two more links, which brings to a cost increase but still guarantees the service availability for all flows.

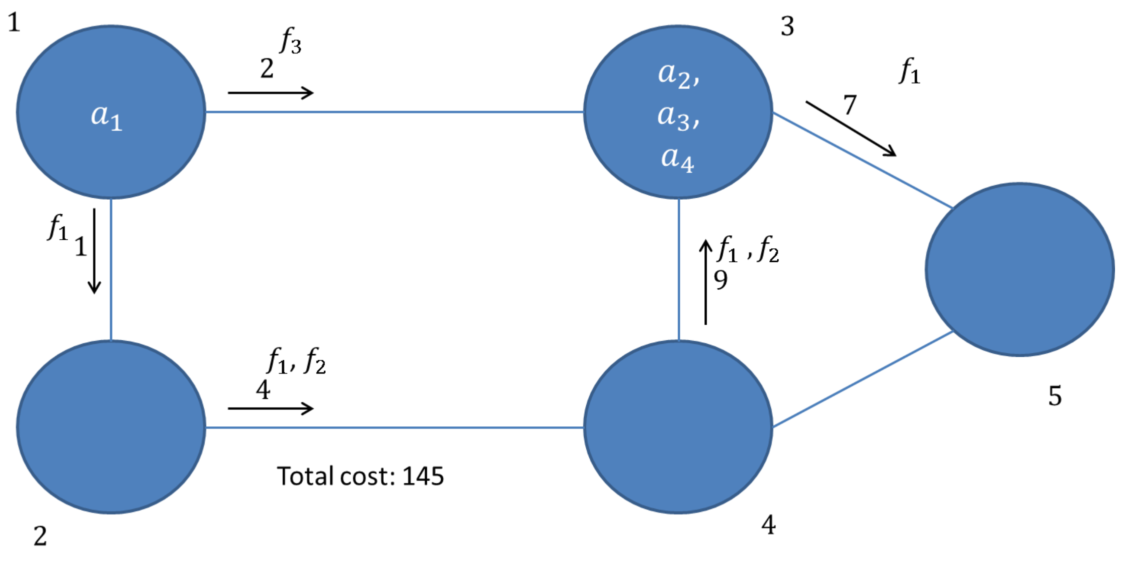


Figure 16. Flows and actions allocation in presence of an increased flow load.

Figure represents a situation in which the activation of Action 1 on Router 3 has a higher cost. In this case, it is possible to activate this action on another router (Router 5) and maintain the network cost constant.

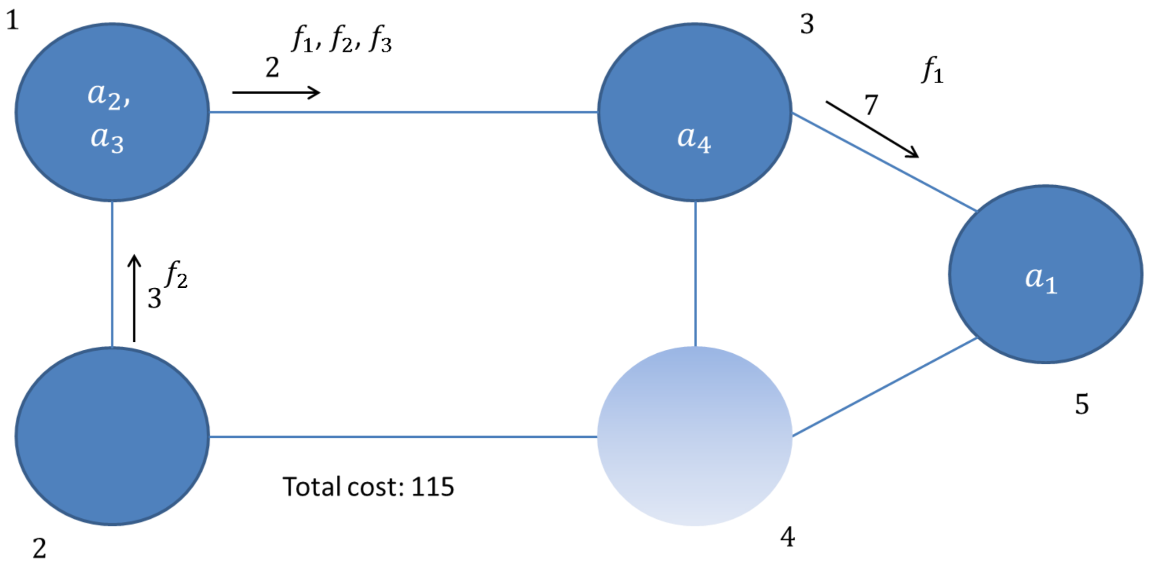


Figure 17. Flows and actions allocation in presence of an increased action cost.

Finally, in the example depicted in Figure , f2 has been removed from the traffic demand. This allows switching off a link and another router to increase the overall energy saving.

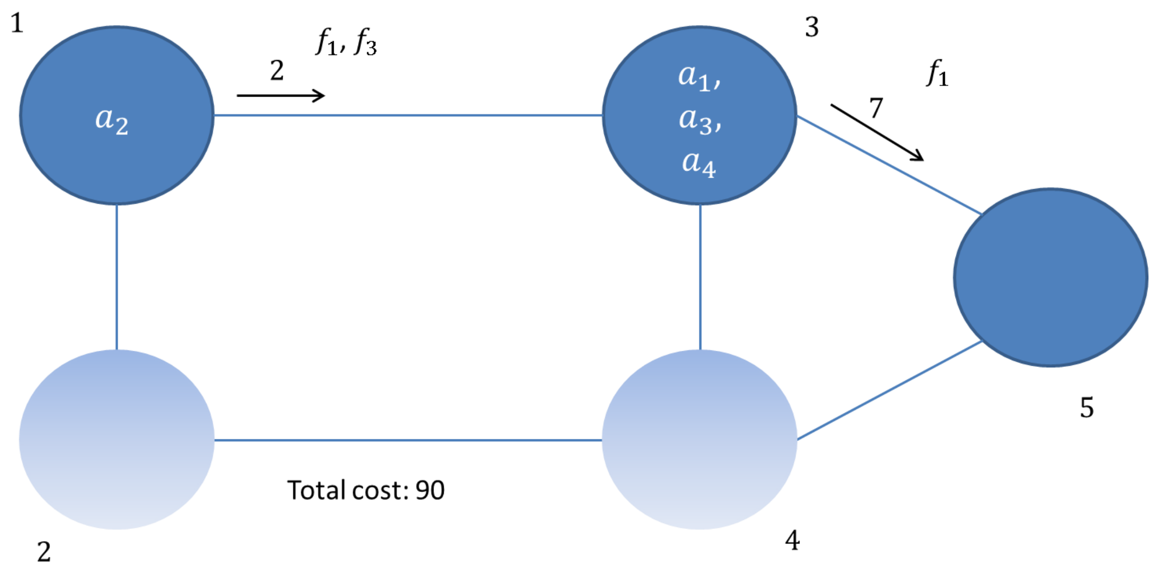


Figure 18. Flows and actions allocation according to a new traffic demand.

6 Network Connectivity Proxy

6.1 Overview

The Network Connectivity Proxy (NCP) maintains the virtual presence for high power devices during their idle periods and wakes them up only when their resources are required [18] The NCP functionality is requested by any low power managed device that wants to reduce its idle energy waste and conveys its virtual presence to another entity (running the NCP) in the local network [19], [20], [21]. The NCP is the combination of the UPnP device as well as control point. As control point, it discovers the low power managed devices (e.g., PCs) in the network and controls them. As UPnP device, it offers the NCP service, advertises its presence in the network and respond to the actions from low power managed devices control point. The basic functionalities of the NCP include:

- Receives discovery messages from the power aware devices and stays up-to-date about their operational power status.
- Sends discovery messages to announce its NCP service to the low power managed devices in the local network.
- Sends action requests to the low power managed devices to control their operational status.
- Receives action requests from the low power managed devices to register different type of proxying requests and wake-up conditions.
- Receives state variable updates from low power managed device and enables/disables the registered actions/rules.
- Sends the Wake-On-LAN (WOL) packet to the sleeping devices when certain wake-up conditions are met.

Apart maintaining network presence, the NCP could also manage the power state of devices, so to put them to sleep when no traffic is present and to wake them up when their processing is required [22]. Thus, the low power managed device is also the combination of the UPnP device as well as control point. As control point, it discovers the NCP presence in the network and requests different actions from it. As UPnP device, it offers the LowPower (LP) service [23], [24], advertises its presence in the network and respond to the actions from the NCP control point. Thus, both the NCP and low power managed device play the role of UPnP device as well as control point as depicted in Fig. 18. This design enables the NCP to control the operational status of the low power managed devices.

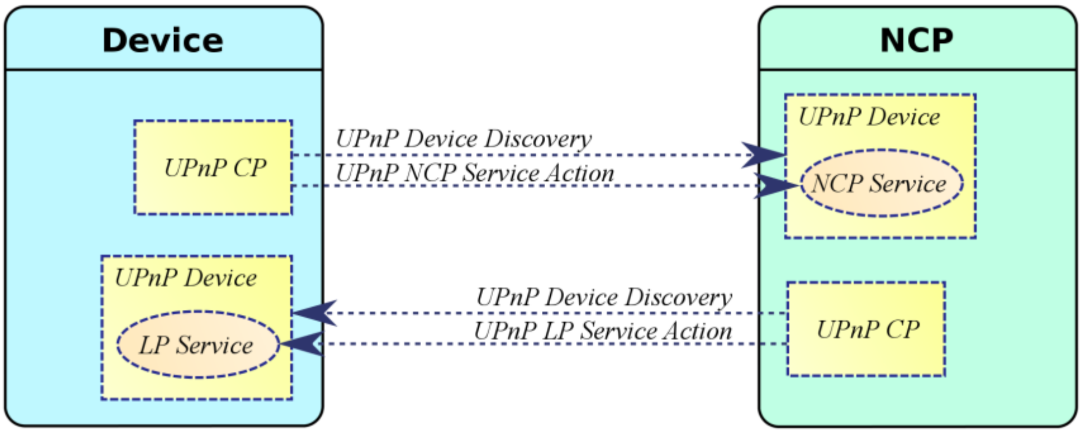


Figure 19. UPnP model integrating NCP and LP services.

6.2 NCP Service

6.2.1 Description

The NCP service is supposed to be running on the residential home gateway. Following the specifications of UPnP Device Architecture v1.0 [17], the basic NCP device and service description is shown below. NetworkConnectionProxy1.xml defines all the actions supported by the NetworkConnectionProxy service (or simply NCP service) and the state variables/action parameters. State variables represent the current state of the UPnP device. The state variables also act as arguments to action definitions. Although the NCP service state variables are only intended to be used as parameters for the actions provided by the NCP service. Thus, such state variables do not really represent the NCP device state and cannot trigger any events. UPnP uses string data type. Actions are send/received in XML format that contain action arguments/parameters as strings. Thus, action arguments have some restrictions and should be converted to string format to represent well formatted XML fragment.

NCP Device Description

```
<?xml version="1.0" ?>
  root xmlns="urn:schemas-upnp-org:device-1-0" configId="1">
        <specVersion>
            <major>1</major>
            <minor>0</minor>
        </specVersion>
        <device>
            <deviceType>urn:schemas-upnp-org:device:NetworkConnectionProxy:1</deviceType>
            <friendlyName>Jetway Atom Home Gateway</friendlyName>
            <manufacturer>Jetway</manufacturer>
            <manufacturerURL>http://www.jetway.com.tw/</manufacturerURL>
            <modelDescription>Jetway Mini ITX with Atom N550</modelDescription>
            <modelName>Jetway JNC9C-550-LF</modelName>
            <modelNumber>1.0</modelNumber>
            <UDN>uuid:03882606-6412-4e01-bfa0-c3755b091a5e</UDN>
            <serviceList>
              <service>
                <serviceType>urn:schemas-upnp-org:service:NetworkConnectionProxy:1</serviceType>
                <serviceId>urn:upnp-org:serviceId:NetworkConnectionProxy:1</serviceId>
                <SCPDURL>/NetworkConnectionProxy1.xml</SCPDURL>
                <controlURL>/control</controlURL>
                <eventSubURL>/eventing</eventSubURL>
       </service>
          </serviceList>
     </device>
  </root>
```

6.2.2 Supported Actions

The NCP service provides a list of actions to the low power managed devices. Any action registration by the NCP service requires host identification and power state information. The host identification is based on the UUID, which is unique to each UPnP device and is supplied by the power managed device during action registration. The power state defines when to activate/execute a particular registered action. Since NCP maintains the presence on behalf of sleeping devices, it responds to ARP packets as soon as the power managed device enters into low power state. The ARP rule is automatically activated for sleeping devices and don't require any registration through UPnP interface. The power managed device can request the following actions from the NCP service.

6.2.2.1 PingRequest Action

The low power managed devices can request the PingRequest action from the NCP service. This action enables the NCP to answer ICMP PING packets on-behalf of power managed devices when they enter into sleep mode.

6.2.2.2 DhcpRequest Action

The low power managed devices can request the DhcpRequest action from the NCP service. This action enables the NCP to periodically renew the IP address of low power managed device with the DHCP server during its sleep period.

6.2.2.3 WakeOnConnection Action

The low power managed devices can request the WakeOnConnection action from the NCP service. This action enables the NCP to wake-up the low power managed device whenever a new connection/packet arrives at specific transport protocol and port.

6.2.2.4 WakeOnPacket Action

The low power managed devices can request the WakeOnPacket action from the NCP service. This action enables the NCP to wake-up the low power managed device whenever a packet matching the supplied pattern is seen. The pattern is supplied by the low power managed device and consists of the data to be looked within the packet with a given offset from the end.

6.2.2.5 SendReplyOnPacket Action

The low power managed devices can request the SendReplyOnPacket action from the NCP service. This action enables the NCP to send a given reply packet on behalf of sleeping low power managed device whenever a packet matching the supplied pattern is seen. The pattern is supplied by the low power managed device and consists of the data to be looked within the packet with a given offset from the end.

6.2.2.6 TcpKeepAlive Action

The low power managed devices can request the TCPKeepAlive action from the NCP service. This action enables the NCP to maintain a given TCP session active on behalf of sleeping low power managed device.

6.2.2.7 HeartBeating Action

The low power managed devices can request the HeartBeating action from the NCP service. This action enables the NCP to send periodic heart-beat messages for the applications used by the power managed devices before sleeping. The application needs to provide all arguments for the HeartBeating action template implemented by the NCP service.

6.2.2.8 TCPMigrationFreeze Action

The low power managed devices can request the TCPMigrationFreeze action from the NCP service. This action freezes the active TCP session at low power managed device and migrates and restores the TCP session at the NCP. The application needs to provide all arguments required for the TCPMigrationFreeze action.

6.2.2.9 TCPMigrationResume Action

The low power managed devices can request the TCPMigrationResume action from the NCP service. This action returns TCP session state from NCP and stores it at low power managed device.

6.2.2.10 Unsubscribe Action

The low power managed devices can request to Unsubscribe an action at the NCP service. The Unsubscribe action enables the NCP to de-register the previously subscribed action.

6.2.3 Integration of UPnP Actions & NCP Rules

The NCP implements several rules to maintain network presence of its sleeping clients by acting on their behalf. This operation implies three main issues:

R1 knowing the power state of devices the NCP is covering;

R2 being aware of traffic addressed to suspended clients;

R3 representing, maintaining and exchanging service state for various protocols and applications.

To cope with the above issues, the NCP must process packets addressed to sleeping/suspended clients and carry out specific tasks based on the registered rules by its clients. Several architectural components are required to build NCP as depicted in Figure 19. The detailed description of the rules is provided in the ECONET project report 49[25]. NCP rules pilot the NCP operations; they bind network traffic addressed towards sleeping clients to specific actions. NCP rules can be divided into three categories; (i) wake-up conditions, (ii) network level presence (e.g., network protocols), (iii) application level presence. Wake-up mechanisms specify different wake-up methods for waking up specific sleeping client. Network traffic filter is an important component of the NCP that sniff, identify and process the packets intended for sleeping clients. Packet identification is normally based on the header information and/or packet content. The actions taken by NCP include; (i) buffer packets, wake-up the sleeping client and send buffered packets after wake-up, (ii) answer packets on-behalf of sleeping client, (iii) send periodic protocols/applications heartbeat messages on behalf of sleeping clients. A communication protocol is required for information exchange and actions registration between the NCP and its clients, which is based on the UPnP protocol as depicted in Figure 19. As described before, both NCP and its clients request actions from one another; NCP requests actions for managing/controlling the power state of its clients, NCP clients request actions from NCP for different kinds of presence proxying requests. Thus, both NCP and its client implements UPnP device as well as control point. Each UPnP action at NCP is linked with the respective NCP rules. NCP rules provide the real implementation of different proxying requests. NCP device state variables do not really represent the device status but are passed as arguments to UPnP actions, which are then subsequently passed as arguments to the NCP rules.

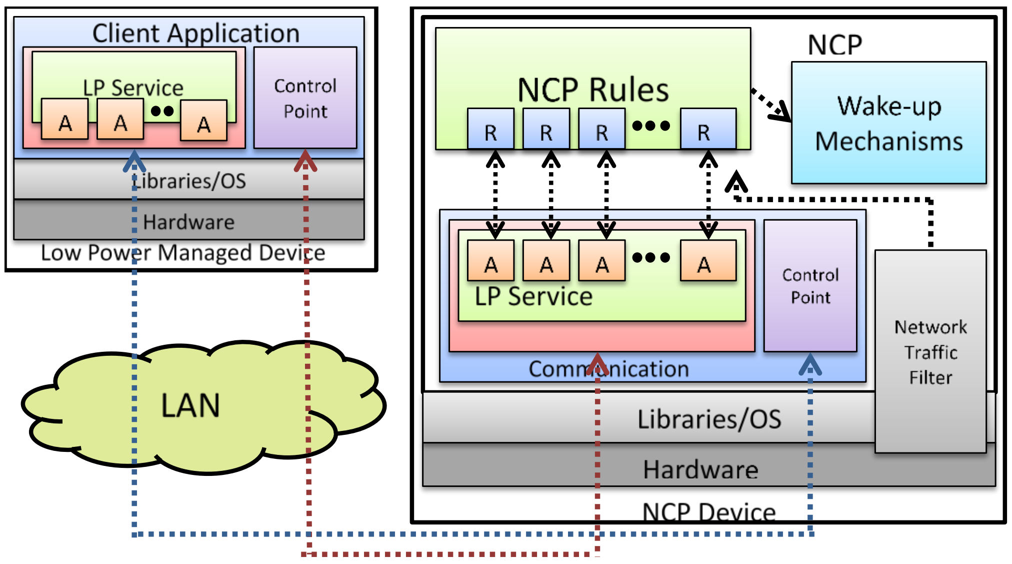


Figure 19. Integration of UPnP actions and NCP rules.

6.3 LowPower Service

6.3.1 Description

The LP service implements the power saving functionalities and is supposed to be provided by the low power managed devices. The detailed description of LP service is provided in UPnP Forum LP Device specification [23], [24]. This service enables the NCP to control the operational status of such devices. Each low power device is supposed to implement one or more sleep states. The low power device may enter into sleep state either autonomously by inactivity timer expiry or can be instructed by NCP. Table 3 shows a brief summary of UPnP power states and the link with those defined for the NCP. A detached bearer means the media is physically disconnected and may require some external human intervention to wake up the device. Finally, note that Table 3 maps UPnP power states into NCP power states, but the reverse relationship does not hold directly.

Following the specifications of UPnP Device Architecture v1.0, the basic LP device and service description is shown below. LowPowerDevice1.xml defines all the actions supported by the LP service and the state variables/action parameters.

Table 3. UPnP power states.

UPnP Power State	UPnP Operation	Bearer operation	Wake up mechanism	NCP Power State
Active	Full	Normal	None	Full power
Transparent Sleep	Full	Power saving	UPnP Wake Up action Autonomous	Low power enabled
Deep Sleep Online	Partial	Power saving	UPnP Wake Up action	Low power enabled
Deep Sleep Offline	Off	Off	Bearer-specific Non bearer-specific Autonomous	Standby
Disconnect	Off	Detached	External	Power off

LP Device Description

```
<?xml version="1.0" ?>
  root xmlns="urn:schemas-upnp-org:device-1-0" configId="1">
       <specVersion>
           <major>1</major>
           <minor>0</minor>
       </specVersion>
       <device>
           <deviceType>urn:schemas-upnp-org:device:NetworkConnectionProxy:1</deviceType>
           <friendlyName>Jetway Atom Set-Top Box</friendlyName>
           <manufacturer>Jetway</manufacturer>
           <manufacturerURL>http://www.jetway.com.tw/</manufacturerURL>
           <modelDescription>Jetway Mini ITX with Atom N550</modelDescription>
           <modelName>Jetway JNC9C-550-LF</modelName>
           <modelNumber>1.0</modelNumber>
           <UDN> uuid:35681486-e0af-4a11-a4e8-180ade25f82b</UDN>
           <serviceList>
             <service>
               <serviceType>urn:schemas-upnp-org:service:PowerManagement:1</serviceType>
               <serviceId>urn:upnp-org:serviceId:PowerManagement:1</serviceId>
               <SCPDURL>/LowPowerDevice1.xml</SCPDURL>
               <controlURL>/control</controlURL>
               <eventSubURL>/eventing</eventSubURL>
       </service>
         </serviceList>
     </device>
  </root>
```

6.3.2 State Variables

The state variables are the reflection of the operational status of the low power managed devices. Detailed description of state variables is provided in UPnP Forum LP service template [25]. The low power devices keep NCP updated about their status through these state variables. The LP service state variables are briefly described in Table 4.

Table 4 LP device state variables.

State variable	Type	Description
ExternalPowerSupplySource	Integer	This says whether the device is powered by an external power supply or by batteries.
BatteryLow	Boolean	Indicate a low battery condition.
PowerSupplyStatus	String	The description of power supplies status, both external and internal (connected/disconnected, battery charge, battery lifetime, etc.).
SleepPeriod	Integer	The sleeping period before the device autonomously wakes up.
SleepPeriodStart	Integer	This indicates the start of sleep period. It is useful if the device is expected to sleep during the specific period of the day.
SleepPeriodEnd	Integer	This indicates the end of sleep period. It is useful if the device is expected to sleep during the specific period of the day.
PowerState	String	Power state of the device.
WakeUpMethod	String	Bearer-dependent method to wake up the device.

6.3.3 Supported Actions

The LP service provides a list of actions to the NCP. These actions enable NCP to manage/control the power state of the low power managed devices. Current actions provided by the LP service are briefly described below.

6.3.3.1 GetPowerManagementInfo Action

The NCP can request the GetPowerManagementInfo action from the LP service. The low power managed device returns power management related information in response to this action. This action enables the NCP to know the power supply status and procedure to wakeup the low power managed device.

6.3.3.2 Wakeup Action

This action enables the NCP to wake-up the low power managed device. The low power managed device provides notification to the NCP after successful wakeup along with the change of PowerState state variable.

6.3.3.3 GoToSleep Action

The NCP can request the GoToSleep action from the LP service. This action provides recommendation to the low power managed device to enter into specified sleep state for the specified period. The low power managed device may decline to sleep if it is busy or performing some operations.

6.3.3.4 DefineSleepPeriod Action

The NCP can request the DefineSleepPeriod action from the LP service. This action provides recommendation to the low power managed device to enter into specified sleep state during the specified period of the day. This action does not provide any output arguments as the device is expected to sleep at some later time. However, the updates about PowerState, SleepPeriodStart and SleepPeriodEnd state variables are provided to NCP whenever the low power managed device enters into sleep mode or wakes up. The low power managed device may decline to sleep if it is busy or performing some operations at the recommended sleep period start time.

6.4 Conclusions and further work

With the introduction of GAL standard interface and related methods, ECONET approach for power management in Telecommunication Networks is based on Protocol extensions that allows smooth introduction of the new technologies preserving compatibility with legacy network.

Protocols can actually be used at different layers, starting from local protocol for proxy servers, to network control plane based on GMPLS PCC/PCE framework, taking also in consideration the current evolution towards NFV and SDN paradigms.

GAL adaptation to different network management technology has been considered, demonstrating the opens and flexibility of the chosen ECONET approach.

More specifically, the proposed Control Plane extension to carry Power Object information across the network, according to the ECONET GAL syntax is oriented to integrate in a simple way energy aware optimization algorithms, as the ones reported in D5.3. The proposal is simple, but concretely effective in making the energy aware routing techniques concretely applicable.

The Network Power Monitoring framework, developing adaptation layers between the standard GAL interface and common management interfaces (such as Rest/XML, CORBA/TMF 814 and Broadband Forum TR-069/CWMP) is essential for extending the usage of the GAL interface outside a single physical object and then for exploiting its functionalities in complex distributed equipment and at the network level. These proposed extensions involve the most used management protocols and they contribute to make the GAL adoption a feasible task in a very large number of situations and scenarios.

The Green extension of Openflow, enabling the usage of the GAL interface and methods into the emerging SDN paradigm has a twofold value: from one side, it projects the GAL in a very hot new network paradigm, SDN (effectively enabling its usage in future advanced equipment based on this approach); on the other side, it defines a very innovate way to extend the current Openflow protocol and architecture to handle the power management capabilities of the SDN devices.

Finally, the Network Connectivity Proxy protocol has been introduced as a key asset of ECONET technology. It is based on a suitable extension of the UPnP standard protocol. Here again, an important work that, if concretely adopted, contributes in a valuable manner to make NCPs easier to introduce in most of the SOHO contexts.

References

[1] The ECONET project, “Final Design of Green Technologies for Network Device Data Plane”, Deliverable 3.3, available on line https://www.econet-project.eu.

[2] The ECONET project, “Definition of energy-aware states”, Deliverable 4.1, available on line https://www.econet-project.eu.

[3] The ECONET project, “Standard interface definitions”, Deliverable 4.2, available on line https://www.econet-project.eu.

[4] The ECONET project, “Abstraction layer final definition”, Deliverable 4.3, available on line https://www.econet-project.eu.

[5] The ECONET project, “Abstract layer prototypic implementation”, Deliverable 4.4, available on line https://www.econet-project.eu.

[6] R. Bolla, R. Bruschi, F. Davoli, L. Di Gregorio, P. Donadio, L. Fialho, M. Collier, A. Lombardo, D. Reforgiato, T. Szemethy, “The Green Abstraction Layer: A Standard Power Management Interface for Next-Generation Network Devices” IEEE Internet Computing, vol. 17, p. 82-86, 2013

[7] R. Bolla, R. Bruschi, F. Davoli, P. Donadio, L. Fialho, M. Collier, A. Lombardo, D. Reforgiato, T. V. Riccobene Szemethy, “A Northbound Interface for Power Management in Next Generation Network Devices”, IEEE Communications Magazine, in press, Vol. 52, No. 1, Jan. 2014. ISSN: 0163-6804.

[8] A. Farrel, Old Dog Consulting, J.-P. Vasseur, Cisco Systems, Inc., J. Ash, AT&T – “A Path Computation Element (PCE) Based Architecture”, IETF RFC 4655, August 2006, http://ww.ietf.org/rfc/rfc4202.txt.

[9] J. P. Vasseur, Cisco Systems, JL. Le Roux, France Telecom – “Path Computation Element (PCE) Communication Protocol (PCEP)”, IETF RFC 5440, April 2009, http://ww.ietf.org/rfc/rfc5520.txt.

[10] E. Mannie, Ed. – “Generalized Multi-Protocol Switching (GMPLS) Architecture”, (IETF RFC3945, October 2004), http://ww.ietf.org/rfc/rfc3945.txt

[11] The IETF Energy MANagement (EMAN) Working Group, https://datatracker.ietf.org/wg/eman/charter/

[12] ECONET Project, “End-user requirements, technology specifications and benchmarking methodologies”, Deliverable D2.1, available on line https://www.econet-project.eu.

[13] Cisco, “Cisco Visual Networking Index: Forecast and Methodology, 2011-2016,” White paper, May 2012, URL: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360_ns827_Networking_Solutions_White_Paper.html

[14] Global e-Sustainability Initiative (GeSI), SMARTer2020: The Role of ICT in Driving a Sustainable Future, Report. http://gesi.org/SMARTer2020.

[15] http://community.comsoc.org/blogs/alanweissberger/sdn-be-352-billion-market-2018-cyans-blue-orbit-carrier-based-sdn-and-nfv-appl

[16] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, “OpenFlow: Enabling Innovation in Campus Networks”, ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69-74, March 2008.

[17] UPnP Forum. UPnP Device Architecture 1.0. October 15, 2008. Available: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0.pdf.

[18] UPnP Forum. UPnP AV Architecture:1. June 25, 2002.
Available: http://www.upnp.org/specs/av/UPnP-av-AVArchitecture-v1-20020625.pdf.

[19] J. Klamra, M. Olsson, K. Christensen, and B. Nordman, "Design and Implementation of a Power Management Proxy for Universal Plug and Play," Proceedings of the Swedish National Computer Networking Workshop (SNCNW 2005), Halmstad, Sweden, November 23-24, 2005.

[20] R. Khan, R. Bolla, M. Repetto, R. Bruschi, M. Giribaldi, “Smart Proxying for Reducing Network Energy Consumption” in Proceedings of the 2012 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS2012), Genoa, Italy, July 8-11, 2012.

[21] R. Bolla, M. Giribaldi, R. Khan and M. Repetto, "Network Connectivity Proxy: An Optimal Strategy for Reducing Energy Waste in Network Edge Devices” in IEEE 24th Tyrrhenian International Workshop on Digital Communications, Genoa Italy, 23-25 September 2013.

[22] R. Bolla, M. Giribaldi, R. Khan and M. Repetto, "Design and Implementation of Cooperative Network Connectivity Proxy using Universal Plug and Play", in 10th edition of Future Internet Assembly (FIA' 2013), Dublin, Ireland, May 2013.

[23] UPnP Low Power Architecture, Version 1.0, August 28, 2007, vailable at: http://www.upnp.org/specs/lp/UPnP-lp-LPArchitecture-v1.pdf.

[24] UPnP Forum. LowPowerDevice:1 Service Template, Version 1.01, August 28, 2007, available at: http://upnp.org/specs/lp/UPnP-lp-LowPower-v1-Service.pdf.

[25] The ECONET project, “Network Connectivity Proxy: Maintaining Network Connectivity for sleeping hosts”, Internal Report n. 7 annexed to the “Final Design of Green Technologies for Network Device Data Plane”, Deliverable 3.3, available on line https://www.econet-project.eu.