ETSI ES 203 237 V1.1.1 (2014-03)

Environmental Engineering (EE);
Green Abstraction Layer (GAL);
Power management capabilities of the future energy telecommunication fixed network nodes

Reference
DES/EE-0030

Keywords
energy efficiency, network monitoring

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://ipr.etsi.org).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This ETSI Standard (ES) has been produced by ETSI Technical Committee Environmental Engineering (EE).

Introduction

Energy efficiency has become one of the most important aspects for both today's and tomorrow's telecommunications infrastructures. The Green Abstraction Layer (GAL) is an architectural interface/middleware that will give flexible access to the power management capabilities of future energy-aware telecommunication fixed network nodes. The present document defines the GAL design interface with the purpose of effectively exploiting the capability of adapting the energy consumption of the network nodes with respect to the load variations.

The main objective is to standardize the interface between the network control processes specifically designed for energy efficiency purposes and the power modulation capabilities of network devices. In particular, the GAL considers three principal aspects:

- making explicit the trade-off between energy consumption and network performance;
- mapping the power consumption of the hardware blocks with virtual and logical network resources;
- hiding the details and complexity of internal power modulation mechanisms.

In this respect, the GAL is composed of two main parts: the Green Standard Interface (GSI) and the Energy Aware States (EASes). The GSI exchanges power management data among data-plane elements and processes realizing control plane strategies in a simplified way. Instead, the EASes describe the different configurations and corresponding performance with respect to the energy consumption of the devices.

The GAL standardization process in ETSI represents a good example of collaboration between ETSI and a European Research Project, because the ideas and the structure of the GAL have been developed in the EU ECONET project [i.1], which is still directly supporting the standardization process in ETSI.

1 Scope

The present document describes the Green Abstraction Layer (GAL), which defines a novel way for managing and monitoring energy and performance profiles of device hardware.

The GAL is an interface between data and control planes for exchanging data regarding the power status of a device. This interface if specifically conceived to hide the implementation details of energy-saving approaches, as well to provide methodologies for interactions between heterogeneous "green" capabilities and Hardware (HW) technologies, on one hand, and control and monitoring frameworks, on the other hand. With "green" capabilities, we refer to any type of mechanisms that implement appropriate optimization policies aimed at reducing the power consumption of a resource. Indeed, this interface provides flexible access to the power management capabilities of the future energy aware telecommunication fixed network nodes to effectively adapt the energy consumption of the network nodes with respect to the load variations.

By means of the GAL, control applications will be 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 round, control applications will be capable of setting a certain power management configuration to the device data-plane. It is worth nothing that, in order to provide the information needed by control applications to reduce the energy consumption while meeting the Quality of Service (QoS) constraints, the GAL explicitly represents the impact on network performance when different power management settings are applied.

The main objective of the GAL is to standardize the interface between the Network Control Policies (NCPs, for the energy efficiency purpose) and the power modulation capabilities of network devices.
There are three key aspects that have to be addressed by the GAL:

a) to make explicit the trade-off between consumption and network performance;

b) to map the consumption of hardware blocks with virtual/logical network resources; and

c) to hide the details/complexity of internal power modulation mechanisms.

In this respect, the present document contains:

- the definition of the GAL general architecture;
- the definition of the interoperable interface (GSI) between the energy-aware control processes (NCPs and LCPs - Local Control Policies) and the power management capabilities of the fixed network devices;
- the definition of the EASes describing the different configurations and corresponding performance with respect to the energy consumption of the devices.

2 References

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

Referenced documents which are not found to be publicly available in the expected location might be found at http://docbox.etsi.org/Reference.

NOTE: While any hyperlinks included in this clause were valid at the time of publication, ETSI cannot guarantee their long term validity.

2.1 Normative references

The following referenced documents are necessary for the application of the present document.

[1] ETSI ES 202 336-1: "Environmental Engineering (EE); Monitoring and Control Interface for Infrastructure Equipment (Power, Cooling and Building Environment Systems used in Telecommunication Networks); Part 1: Generic Interface".

[2] IETF RFC 4122: "A Universally Unique Identifier (UUID) URN Namespace".

NOTE: Available at http://www.ietf.org/rfc/rfc4122.txt.

[3] IETF RFC 4133: "Entity MIB".

NOTE: Available at http://www.ietf.org/rfc/rfc4133.txt.

[4] IETF RFC 3433: "Entity Sensor Management Information Base".

NOTE: Available at http://www.ietf.org/rfc/rfc3433.txt.

[5] IEC 61850: "Power utility automation".

NOTE: Available at http://www.iec.ch/smartgrid/standards/.

2.2 Informative references

The following referenced documents are not necessary for the application of the present document, but they assist the user with regard to a particular subject area.

[i.1] ECONET (low Energy COnsumption NETworks) large-scale Integrating research Project (IP), funded by the European Commission under the 7th framework programme, theme ICT, call 5, grant agreement no. 258454.

NOTE: Available at http://www.econet-project.eu.

[i.2] COMMISSION REGULATION (EC) No 1275/2008 of 17 December 2008 implementing Directive 2005/32/EC of the European Parliament and of the Council with regard to ecodesign requirements for standby and off mode electric power consumption of electrical and electronic household and office equipment.

NOTE: Available at http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2008:339:0045:0052:EN:PDF.

[i.3] IETF Energy MANagement (EMAN) Working Group: "Energy Management (EMAN)".

NOTE: Available at https://datatracker.ietf.org/wg/eman/charter/.

[i.4] B. Claise and J. Parello: "EMAN: Energy-Management Activities at the IETF", IEEE Internet Computing, vol. 17, no. 3, pp. 80-82, May-June 2013.

[i.5] Advanced Configuration & Power Interface (ACPI).

NOTE: Available at http://www.acpi.info.

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Advanced Configuration and Power Interface (ACPI): interface providing an open industrial standard for device configuration and power management by the operating system

NOTE: ACPI defines platform-independent interfaces for hardware discovery, configuration, power management and monitoring. The specification is central to Operating System-directed configuration and Power Management (OSPM), a system implementing ACPI, which removes device management responsibilities from legacy firmware interfaces.
ACPI aims to consolidate, check and improve upon existing power and configuration standards for hardware devices. This specification contains numerous related components for hardware and software programming, as well as a unified standard for device/power interaction and for bus configuration.
It requires that once an OSPM-compatible operating system has activated ACPI on a computer, it then takes over and has exclusive control of all aspects of power management and device configuration. The OSPM implementation exposes an ACPI-compatible environment to device drivers.

Adaptive Rate (AR): technique to dynamically modulate the capacity of a network device (or a sub-component) in order to meet traffic loads and service requirements

Convergence Layer Interface (CLI): GAL interface designed to map the GAL commands and data into low-level configuration registers/APIs, which are often manufacturer/HW specific

Energy-Aware Entity (EAE): entity that can trade its power consumption and performance, available functionalities, and responsiveness

Energy-Aware State (EAS): entity power setting managed through the GAL that impact on the power consumption, the performance, the available functionalities, and the responsiveness of the entity

NOTE: An EAS can be considered as an operational power profile mode implemented by the entity that can be configured by control plane processes. To assure the correct exchange of information between the entity and control plane processes, the EAS should be represented as a complex data type, which contains indications on the power consumption, the power saving, the performance, the available functionalities, and the responsiveness of the entity when working in such configuration.

Entity: device or a sub-part of it, of which the GAL constitutes the energy-aware interface

NOTE: At the lowest hierarchical levels, an Entity can correspond to a chip, a network processor, a link interface. At medium hierarchical levels, it can correspond to line-cards, chassis, etc. At the highest level the entire device corresponds to an entity. Higher level entities can include one or more entities at lower levels. This hierarchical architecture is optional and the relative depth should depend on the specific internal architecture of the network device. The terms "entity" and "resource" are used interchangeably in the present document.

Entity Dependency List (EDL): optional structure meant to convey read-only information regarding which further EAS settings could help in achieving significant power saving

Green Abstraction Layer (GAL): interface between data and control planes for exchanging data regarding the power status of a device

Green Standard Interface (GSI): GAL interface designed to exchange power management data in a simplified way among data-plane elements and processes realizing control plane strategies

Local Control Policy (LCP): control process to optimize the configuration at the device level, in order to achieve the desired trade-off between energy consumption and network performance, according to the incoming traffic load

Low Power Idle (LPI): technique to force a network device (or a sub-component) to enter low power states when it does not forward/process packets (idle operating mode)

Monitoring and Operation Administration & Management: processes, activities, tools, standards, etc., involved in operating, administering, managing and maintaining a system

Network Control Policy (NCP): control process with the aim to optimize the behaviour (through the tuning of the trade-off between power consumption and performance) of a network (i.e. a set of interconnected devices)

operating mode: operating state of a given entity (i.e. *active*, *standby* - or *idle*)

NOTE: The operating state *standby* and *idle* are considered interchangeable. It is required to distinguish two different types of EAE:

- with *automatic (internal) operating mode transition* - when the EAE changes the operating state automatically. In this case, the setting of a specific EAS consists exclusively of a configuration of the trade-off between energy saving and network performance;
- with *manual (external) operating mode transition* - when the change of the operating state is controlled by external processes (e.g. LCPs or NCPs). In this case the setting of a specific EAS does not assert only the configuration of the trade-off between power saving and network performance, but it dictates also the change of the operating mode.
  The CLI is responsible to implement the transitions in a transparent manner to the GSI. Indeed, the GSI manages only the EASes.

standby mode: operating mode characterized by low power consumption and reduced functionality

NOTE: This mode saves significantly on power consumption compared to leaving an entity in active mode. The reduction can be done by cutting power to unused entity components. For this reason, in standby mode, the entity provides a sub-set of functionality depending on the specific energy profile. This notion of standby is different from the one defined in the European Commission Regulation 1275/2008 [i.2] where the functions provided regard only the reactivation and the information display.

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ACPI	Advanced Configuration and Power Interface
API	Application Program Interface
AR	Adaptive Rate
CLI	Convergence Layer Interface
EAE	Energy Aware Entity
EAS	Energy Aware State
ECONET	low Energy COnsumption NETworks
GAL	Green Abstraction Layer
GSI	Green Standard Interface
HW	Hardware
IEC	International Electrotechnical Commission
LAN	Local Area Network
LCP	Local Control Policy
LPI	Low Power Idle
NCP	Network Control Policy
NMS	Network Management System
OAM	Operations, Administration & Management
OSPF-TE	Open Shortest Path First - Traffic Engineering
PDU	Protocol Data Unit
P-PsS	Power scaling Primitive sub-State
PsS	Primitive sub-State
QoS	Quality of Service
RFC	Request For Comments
RSVP-TE	Resource Reservation Protocol - Traffic Engineering
SI	International System of Units (abbreviated SI from French: Le Système international d'unités)
S-PsS	Standby Primitive sub-State

SW	Software
WoL	Wake on LAN (remote wake-up of client machine by the network)
WoP	Wake on Packet

4 Green Abstraction Layer

The GAL synthetizes the data related to power management settings into a sort of standard data objects, namely, "energy-aware states" (EASes). The GAL manages the EASes by means of two interfaces:

- Green Standard Interface (GSI) - to exchange power management data in a simplified way among data-plane elements and processes realizing control plane strategies;
- Convergence Layer Interface (CLI) - optional interface to map the GAL commands and data into low-level configuration registers/APIs, which are manufacturer/HW specific.

Since it is up to each specific GAL instance developer to choose how to interact with the particular hardware, the CLI interface is not described in the present document.

In general, the GAL should be used by three main sets of control plane processes:

- Local Control Policies (LCPs), which optimize the configuration at the device level, in order to achieve the desired trade-off between energy consumption and network performance, according to the incoming traffic load. To this purpose, such processes need to know in detail the internal architecture of the device (or parts thereof), the number, the typology and the capability of energy-aware elements, as well as to have access to network performance indexes.
- Network Control Policies (NCPs), to autonomously control and optimize the behaviour of a network. Typical examples of this kind of processes are traffic engineering, routing and signalling algorithms/protocols (e.g. OSPF-TE/RSVP-TE) with "*green*" extensions.
- Monitoring and Operation Administration & Management (OAM), for the operator to control and optimize the trade-off between energy consumption and network performance through a network management system with "*green*" capabilities.

Network-wide policies, monitoring and OAM frameworks are already and largely present in current network technologies and protocols, but nowadays they are almost completely oriented to networking performance, and unaware of the network energy use. Their main objective can be summarized in moving traffic through the network in order to meet end-to-end connectivity, QoS, and resilience constraints, and monitor them within the sanity level of devices. Nevertheless, at the current state of the art there are several proposals that would extend these mechanisms with the aim of taking into account the energy consumption, too. An example of these proposals can be the IETF Energy Management (EMAN) [i.3], [i.4]. The EMAN is oriented towards the collection of detailed data about the consumption and power states of the networked devices. In order to manage different types of networked devices, it cannot provide a measure of the trade-off between performance and power consumption. In this context, the GAL framework can be used to provide a more accurate description of the energy level of the network devices, by adding the achievable performance for a specific energy-state configuration, with the aim of reducing the energy absorption of the whole network, while guaranteeing network performance constraints. In addition, the GAL can provide a simplified view of the various network devices (and their components), by hiding the internal architectural complexity.

In a network consisting of thousands of controlled systems, it cannot expected that a centralized or distributed global algorithm can react to fast changes in network demand with time scales of μs, ms or maybe even seconds. So, it should expected that some "local" control loops will be provided by each system. An example of such control loop can be the automatic link level throttling of the number of active link lanes as a function of the imposed traffic demand. Another example is the automatic shutdown of some of the parallel network processing cores within a network processing engine. At the router system level, an example can be turning off some of the switching elements.

The energy-aware LCPs, which will implement the control loops mentioned above, are a novel concept in the networking field. Today's devices do not include such policies, since each internal element simply works at its maximum speed all of the time, and all the elements in a device are dimensioned to provide a given level of performance.

Therefore, owing to the main effects of energy-adaptation primitives on network performance, local policies are fundamentally needed for:

- dynamically managing the configuration of one or more energy-aware elements in order to meet the overall desired energy savings and performance targets (so that a number of local control strategies can run on the same device by controlling different parts of it - e.g. a process for each line-card);
- guaranteeing the coherence of configurations among different (groups of) energy-aware elements in order to make the entire device working as expected.

To this purpose, local control strategies need to act by accurately knowing the underlying HW elements, the effects of their energy-aware capabilities, and how they are interconnected among themselves. Hence, the information to be used by local policies should be at a more detailed level than the one usually exposed by network-wide and OAM frameworks, which typically represent devices only in terms of links and nodes.

The adoption of local, network-wide and OAM policies should not be exclusive, but they are conceived to work with different goals and contexts in a complementary way. Notwithstanding these differences, the behaviour of local policies cannot be completely independent of network-wide and OAM control frameworks. A simple example of such dependency is given by the fact that the optimal local configuration relies on the traffic load incoming from device links, which, in its turn, is influenced by network-wide and OAM policies.

4.1 GAL Main Design

The specific goal of the GAL is to extend and re-engineer the ACPI standard [i.5] for general purpose computing systems, and adapt it to architectures, functionalities and paradigms of network devices, and especially of their data-plane components.

It is worth noting that the GAL is a device internal interface, so it does not behave as a network signalling protocol. Control-plane processes implement signalling protocols to make more network nodes agree on a certain configuration. This is not a direct goal of the GAL, but these control-plane processes need to be interfaced with the GAL for acquiring/setting power-aware parameters.

In other words, the GAL shall be conceived to enable control processes acquire information on the "green" capabilities available at the data-plane, configure them, and carry measurements on the energy consumption.

Thus, unlike the ACPI standard, the definition of the GAL given in the present document considers:

- the presence and the main features of multiple and heterogeneous internal components (e.g. link interfaces, multiple chips for packet processing, fans, etc.) with energy adaptation and monitoring capabilities as described in ES 202 336-1 [1];
- heterogeneous and modular architectures where these internal components are usually embedded and, consequently, how they can depend and interact among themselves;
- the provision of methodologies for aggregating all the information from single internal components (e.g. single component, link level, line card, chassis and node levels);
- the characterization of the effect of putting components in a certain energy configuration in terms of power consumption and network performance (e.g. minimum and maximum energy consumption, maximum throughput, etc.);
- the definition of Energy-aware States and the information they have to provide.

In addition, the GAL definition includes the following optional aspects:

- a suitable methodology for representing the internal "topology/organization" of the device (how energy aware elements are interconnected and depend among themselves);
- a hierarchical representation of the device internal architecture, in order to support different detail levels on the internal architecture (e.g. single component, link, line card, chassis and node), which can be needed by the considered control strategies;

- the management of the interface between local control policies, which work inside the device and its components at the various hierarchical levels, and the other upper-layer network and OAM control processes, which generally act on top of the control chain and cannot see the internal device peculiarities. It is worth noting that, typically, an upper-layer control process (e.g. traffic engineering) can require to see the effect of the application of a local control policy (in order to take its own decisions), or to set (as a result of its decision) parameters or constraints that determine the operating behaviour of the local control policy.

4.2 GAL Functional Architecture

The GAL framework is a multi-layer hierarchical interface that allows the intercommunication among multiple local and network-wide control planes and the data-plane hardware, with the aim of managing the trade-off between device power consumption and network performance. The different layers are meant to represent various abstraction levels of power managed devices. The GAL constitutes the energy-aware interface of *Entities* that can represent whole devices or a part of them. At the lowest hierarchical levels, an Entity can correspond e.g. to a processor, a link interface, or a fan. At medium hierarchical levels, it can correspond to line-cards, chassis, etc. Finally, at the highest level the entire device corresponds to an entity. Higher level entities can include one or more entities at lower levels. The hierarchy depth depends on the specific internal architecture of the network device. In particular, we refer to Entities with power saving capabilities as Energy-Aware Entities (EAEs). The hierarchical architecture should not always be necessary, so it is considered optional; only the top layer shall be mandatory, in order to allow the communication between the network device elements and the control processes (LCPs and NCPs).

In summary, the main goals of the GAL architecture are:

- to provide a structured methodology to manage the EASes of GAL-compliant devices;
- to hide the hardware architecture complexity to the control plane processes;
- to enable LCPs to interact with control-planes and EAEs by means of standard high-level commands (passed through the GSI).

To achieve these goals, the GAL framework includes two main interfaces: GSI and CLI. The first one represents the standard part of the GAL, whereas the latter may map GAL commands and data onto low-level configuration registers/APIs, which are manufacturer/HW specific.

Figure 1 shows an example with 4 levels of different granularity (from device level to HW-component level through the chassis and line-card ones). The control processes interact with the EAEs at different levels by means of the GSI. Then, the commands are sent to the HW by means of the CLI. Only the bottom layer communicates with the Convergence Layer.

As previously highlightened, the hierarchical structure shall not be mandatory, and the present document does not define how many and which levels should be defined. For more information on the GAL hierarchical structure see Annex A.

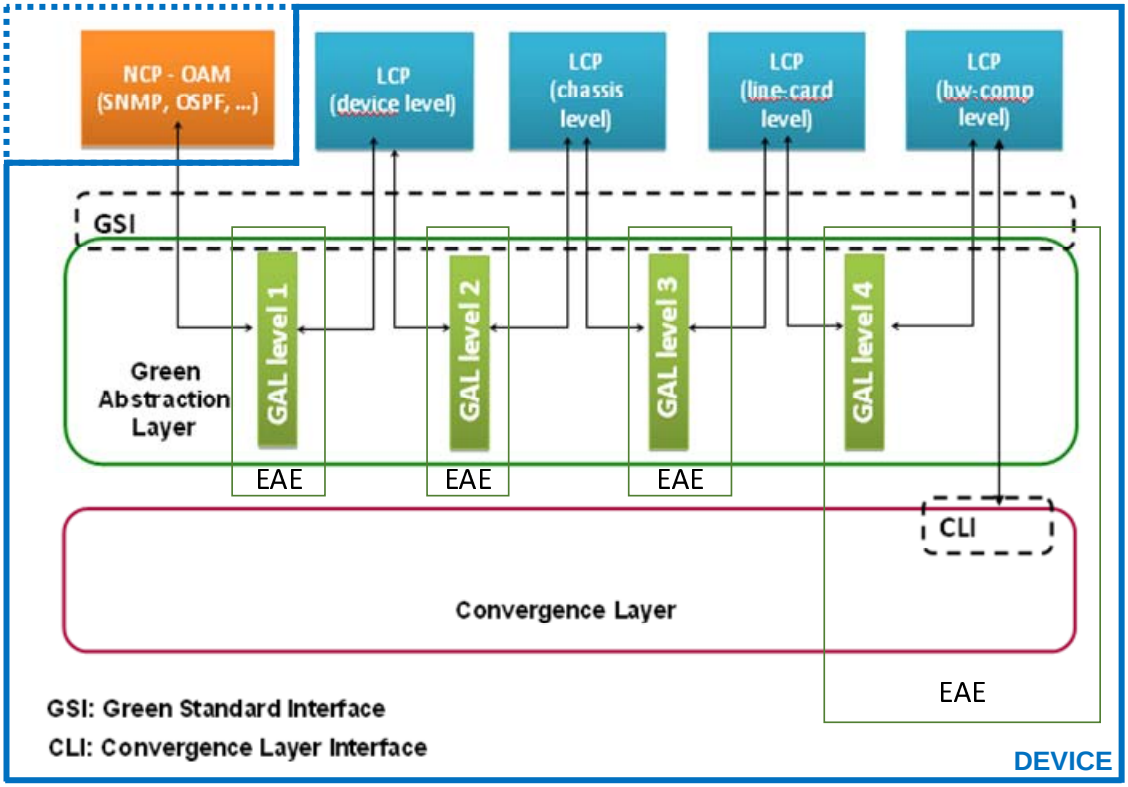

Figure 1: GAL architecture

Finally, the current definition of the GAL architecture suggests some guidelines which, although optional, simplify the interaction with it. To completely exploit the abstraction provided by the GAL and avoid interacting with the architectural complexity of the devices all the communications should be made through the LCPs (top-level or internal).

5 Green Standard Interface (GSI)

The Green Standard Interface (GSI) is the northbound interface of a generic GAL instantiation. It is in charge of providing the command set necessary to setup the power management and monitoring of a wide set of energy-aware resources and network devices. In other words, the GSI is used for the access to resources and their power characteristics discovery, the autonomic provisioning and manual configuration of resources, and the monitoring and the decommissioning of energy-aware physical resources as described in ES 202 336-1 [1].

The GSI methods are designed to manage a single command (i.e., only one configuration change request per command can be supported). Bulk commands are not supported. However, the GSI supports "commit/rollback" methods, which allow exploring different complex configurations (e.g. coming from multiple EAS requests), before really applying them into the device HW. Every time an EAS change command is sent through the GAL, the overall performance indexes and the available EASes of the affected entities are updated accordingly.

Although the exact implementation will depend on the devices (e.g. network elements, network cards, chips, fans, etc.), it includes a mandatory set of messages and workflows defining the abstract messages to enable the GAL interface operation. The API messages offered by the GSI are listed in Table 1. They manage the trade-off between power consumption and network performance representing the EAS with the *resource* structure. There are two different types of resources: *physical* and *logical*.

The former represent hardware entities, while the latter are an aggregation of one or more resources (physical or logical) that provide a summarized and simplified view of the managed entity to the control/management planes or to the LCPs. A high level example of logical resource can be a router represented by a set of several physical/logical resources (e.g. network processors, interfaces, controllers, buffers, etc.). Each resource can contain physical and/or logical components depending on the specific implemented hierarchical structure.

Table 1: GSI API messages

Functionality	Direction	Description
Discovery	Information request GAL client ⇨ EAE	Request to retrieve information about available power states and other descriptive information of the entity.
	Information response GAL client ⇦ EAE	Response with information about the entity and the list of individually manageable components (both logical and physical).
Provisioning	Configuration command GAL client ⇨ EAE	Request to configure an entity into a certain power state.
	Configuration notification GAL client ⇦ EAE	It returns the operating results: *error* or *success*.
Release	Decommissioning command GAL client ⇨ EAE	Request to release the committed device resources.
	Decommissioning notification GAL client ⇦ EAE	Notification of device resource release.
Monitoring	Parameter request GAL client ⇨ EAE	Request to retrieve relevant information about the status and energy consumption of the entity.
	Parameter notification GAL client ⇦ EAE	Notification about relevant entity parameter changes (status, energy consumption).
Commit	Confirmation command GAL client ⇨ EAE	Request to make changes done in a provisioning request permanent.
	Confirmation notification GAL client ⇦ EAE	It returns the operating results: *error* or *success*.
Rollback	Rollback command GAL client ⇨ EAE	Request to rollback the state of the GAL to the last commit point.
	Rollback notification GAL client ⇦ EAE	It returns the operating results: *error* or *success*.

The main functionalities offered by the GSI are the following:

- **Discovery:** used to retrieve information about:
  - available power states and other descriptive information of the entity;
  - measurement/monitoring points for reading power-related information;
  - list of individually manageable components within the entity and their relation (both logical and physical).
- **Provisioning:** allows the configuration of an entity into a certain power state.
- **Release:** allows putting the entity in its default configuration.
- **Monitoring:** permits to monitor relevant parameters (state, power consumption, temperature, etc.) of the physical device.
- **Commit:** permits to confirm the changes done in the provisioning request.
  In the GAL architecture only the GSI (and its implementing sub-services) have the information necessary to estimate actual power consumption for a given entity configuration, where different entity components are in different power states, managed by external controlling entities (NCPs), internal entity constraints, etc.
  In this scenario, the GSI distinguishes two types of parameters: one for the current operation mode/operation parameters setting and the one/ones to be activated. The distinct steps are synthesized as follows:
  1) the controlling agent (NCP/higher level LCP) sets the next state (Provisioning State), even if the entity does not yet physically enter the state;
  2) the controlling agent (NCP/higher level LCP) can query power consumption data "as if the entity were in Provisioning State mode";
  3) the controlling agent can commit the entity into the mode set in step 1).
- **Rollback:** it shall be the inverse of the Commit command. It undoes some or all GAL changes made during the current transaction.

5.1 GSI API Parameters

The GSI shall be controlled through the following list of input and output parameters:

- **resource_id** (UUID - as defined in RFC 4122 [2]) - input parameter that represents the unique resource identifier of the entity to be managed. It allows managing entities with different granularity both at infrastructure and component level (network, device, interface and logical resources).

- **committed** (BOOLEAN) - input parameter that indicates if the request is done for the current state or for the provisioned state. The possible values are:

  - TRUE - the information obtained regarding the current state.

  - FALSE - the information obtained regarding the provisioned (not committed) state.

- **resource** - output structure that, after a discovery operation, contains the information of the entity to be managed. The data structure shall be the following:

  <*resource_id*, *description*, *type*, *depends_on*>

  where:

  - *resource_id* (STRING) - contains the unique resource identifier;

  - *description* (STRING) - contains the description of the resource;

  - *type* (UINT16) - high-level description of the resource type (e.g. cpu, chassis, module, port, etc.). The GAL refers to *entityPhysicalClass* identifiers defined in RFC 4133 [3] to fill this parameter;

  - *depends_on* (LIST<*resource_id*>) - contains identifiers of the resources that influence the operation of the current resource. A resource can depend on other resources. A change in the configuration of those resources can cause a change in the configuration of the depending resources.

- **logical_resources** (LIST<*resource*>) - output structure that, after a discovery operation, contains the list of available logical resources inside the entity to be managed. The data structure shall be the same of the **resource** parameter.

- **physical_resources** (LIST<*physical_resource*>*)* - output structure that, after a discovery operation, contains the list of available physical resources inside the entity to be managed. The *physical_resource* data structure shall be the following:

  <*resource_id*, *description*, *type*, *used_by*, *depends_on*>

  where:

  - *resource_id*, *description*, *type*, *depends_on* as defined for the **resource** parameter;

  - *used_by* (LIST<*resource_id*>) - contains the identifiers of the physical/logical resources which use the current resource.

  The physical_resources shall be empty if the internal architecture of the device is unknown. In addition, the physical resources in this list shall be referred by at least one resource in the logical_resources list.

- **sensor_resources** (LIST<*sensor_resource*>) - output structure that contains the list of available sensors within the resource after a discovery operation. The *sensor_resource* data structure shall be the following:

  <*sensor_id*, *description*, *type*, *scale*, *precision*, *refresh_rate*>

  where:

  - *sensor_id*, (UUID - as defined in RFC 4122 [2]) - parameter that represents the unique resource identifier of the sensor;

  - *type* corresponds to *entPhySensorType* defined in RFC 3433 [4]. The possible values are: other, unknown, voltsAC, voltsDC, amperes, watts, hertz, celsius, percentRH, rpm, cmm, and truthvalue;

- *scale* corresponds to RFC 3433 [4] *entPhySensorScale*;
- *precision* corresponds to RFC 3433 [4] *entPhySensorPrecision*;
- *refresh_rate* corresponds to RFC 3433 [4] *entPhySensorValueUpdateRate*.

- **eas** - output parameter that represents the Energy Aware State (EAS). See Clause 6 for a detailed description of the EAS.
- **eas_id** (`INTEGER`) - output parameter that identities the EAS. For the identification, it refers to the attribute *n* of the EAS data structure. See Tables 9 to 11 for a detailed description of this and other attributes.
- **eas_list (**`LIST<`*eas*`>)` - output list that contains the available energy aware state of the given entity.
- **power_consumption** (*Measurement*) - output parameter that represents the power consumption of the resource. The default value of the unit attribute of the *Measurement* data type (described in Table 2) should be "mW" (milli-Watt).
- **eas_history** - output parameter that represents the energy aware state evolution of the resource as a list ordered by *time_entered* and represented by the following data structure:

  <*eas_id*, *time_entered*, *power_consumption*>

  where:

  - *eas_id (INTEGER) - energy aware state id;*
  - *time_entered* (`TIMESTAMP`) - time when the entity entered the designated energy aware state;
  - *power_consumption* (`Measurement`) - The value of the unit attribute of the `Measurement` data type should be "mW" (milli-Watt).
- **entity dependency list (ed_list)** - output structure implemented as an ordered array of lists, containing a set of "suggested moves" to obtain significant energy saving. The data structure shall be the following:

  `LIST<`*optimal_configs*>

  where:

  - *optimal_configs* - list of suggested configurations composed by:

    `LIST<`*resource_id*, `LIST<`*eas_id*`>>`

  The *ed_list* structure provides a list of optimal configurations. This list shall be organized by decreasing values of power gains. If some devices cannot suggest an optimal configuration, the *ed_list* could be `NULL`.

The values that relate to power, transition times and network throughput (in terms of bits per second or packets per second) are defined by the `Measurement` data type as show in Table 2.

A measurement shall be qualified with the unit, magnitude (*multiplier* attribute), and should be qualified with caliber, and accuracy when applicable. Measurement magnitude shall conform to the unit multiplier of the SI (International System of Units) units of measure as defined in IEC 61850 [5]. Measured values are represented in SI units obtained by *value* × ($10^{multiplier}$). For example, if the current power consumption is 5, it could be 5 W, 5 mW, 5 kW, or 5 MW, depending on the value of the scaling factor and the unit attributes. 5W implies that the *value* is 5, *multiplier* = 0 and *unit* = "W" whereas 5 mW can imply *value* = 5, *multiplier* = -3, and *unit* = "W" or *value* = 5, *multiplier* = 0, and *unit* = "mW".

Instead, the caliber attribute indicates how the measurements were made at the entity/resource describing the method that was used to measure (`static`, `actual`, or `estimated`).

Finally, the *accuracy* attribute indicates the level of accuracy of the measurement in percentage terms (± *accuracy* % with respect to the measured value).

Table 2: Measurement Data Type attributes

Parameter	Data Type	Optionality	Default Value
value	INTEGER	Mandatory	
unit	STRING	Mandatory	
multiplier	INTEGER	Mandatory	0
caliber	ENUM {static, actual, estimated}	Optional	static
accuracy	PERCENTAGE	Optional	0

5.2 GSI command set

The GSI API set shall be composed by six commands. The next subclauses introduce each GSI command and the related syntax.

5.2.1 GAL_Discovery command

The *GAL_Discovery* permits to retrieve information related to the resource infrastructure. The syntax is shown in Table 3.

Table 3: GAL_Discovery syntax

Parameters	IN	1	resource_id
		2	committed
	OUT	3	resource
		4	logical_resources
		5	physical_resources
		6	sensor_resources
		7	eas_list
		8	ed_list
Return	ENUM{*error, success*}		

5.2.2 GAL_ Provisioning command

The *GAL_Provisioning* permits to configure an individual resource or a group of resources. The syntax is shown in Table 4.

Table 4: GAL_Provisioning syntax

Parameters	IN	1	resource_id
		2	eas_id
Return	ENUM{*error, success*}		

5.2.3 GAL_Monitor command

The *GAL_Monitor* command set permits to monitor the energy parameters and values of an individual resource or a group of resources.

There are three commands to monitor entities:

- *GAL_Monitor_State* - returns the current energy aware state of the resource. The syntax is shown in Table 5;
- *GAL_Monitor_Sensor* - returns the current readings of a sensor resource. The syntax is shown in Table 6;

- *GAL_Monitor_History* - optional command that returns a list representing the eas history of the resource. The syntax is shown in Table 7.

Table 5: GAL_Monitor_State syntax

Parameters	IN	1	resource_id
		2	committed
	OUT	3	eas_id
Return	ENUM{*error, success*}		

Table 6: GAL_Monitor_Sensor syntax

Parameters	IN	1	resource_id	
	OUT	2	oper_status	Defined in RFC 3433 [4]
		3	sensor_value	
		4	value_timestamp	
Return	ENUM{*error, success*}			

Table 7: GAL_Monitor_History syntax

Parameters	IN	1	resource_id	
	OUT	2	eas_history	LIST<TIMESTAMP, eas_id>
Return	ENUM{*error, success*}			

The implementation of this command is optional, and the depth of the list `eas_history` is arbitrary. However, in case a non-empty list is returned, it should be ordered by the time parameter.

5.2.4 Other GAL commands

The last three GAL commands (*GAL_Release, GAL_commit, and GAL_Rollback*) regard the management of the entity configuration.

GAL_Release permits to put the resource in its default configuration. Instead, *GAL_Commit* permits to confirm the changes done in the provisioning request. Finally, *GAL_Rollback* shall be the inverse of the *Commit* command. It undoes all GAL changes made during the provisioning command. In other words the rollback cancels all Provisioning commands issued after the last Commit. The syntax of these commands is shown in Table 8.

Table 8: GAL_Release, GAL_Commit, and GAL_Rollback syntax

Parameters	IN	1	resource_id
Return	ENUM{*error, success*}		

6 Definition of Energy-aware States

An Energy Aware State (EAS) can be modeled as a couple of energy-aware Primitive sub-States (PsSes) related to the configuration of Power Scaling and Standby mechanisms:

$$EAS_n = \{P_j^{(n)}, S_k^{(n)}\} \text{ with } 0 \leq j \leq J \text{ and } 0 \leq k \leq K$$

where:

- $S_k^{(n)}$ represents the *k*-th PsS related to the standby techniques:
  - $S_0^{(n)}$ shall be the active state,

- in $S_k^{(n)}$, with $k > 0$. the entity is sleeping,
- in $S_K^{(n)}$, the device is completely off.

- $P_j^{(n)}$ represents the *j*-th PsS related to the Power Scaling techniques:
  - $P_0^{(n)}$ shall be the PsS for the highest network device performance (it also consumes the highest power),
  - with $j > 0$, $P_j^{(n)}$ shall be the PsS for the intermediate level where the entity is in reduced performance while still being active.

Given the exclusive use of power scaling and standby capabilities, the *J* Power Scaling PsSes (P-PsSes) can be adopted only when the entity is active while the *K* Standby PsSes (S-PsSes) only when the entity is in standby mode.

In this way, when an entity provides internal automatic mechanisms for changing the operating mode (active or standby), the EAS P-PsS component represents the power scaling configuration used during the active mode, while the S-PsS component represents the standby configuration used during the standby mode. Examples of entities that provides these automatic mechanisms are Network Processor Engines, CPUs, etc.

Instead, when an entity does not provide any automatic mechanisms for changing the operating mode, the EASes exposed by this entity will consist of the following allowable combinations:

- $\{P_j^{(n)}, S_0^{(n)}\}$ with $0 < j \leq J$ for EASes representing the power scaling configuration;
- $\{P_0^{(n)}, S_k^{(n)}\}$ with $0 < k \leq K$ for EASes representing the standby configuration.

In these cases, the change of the operation mode is implicitly performed through the setting of a specific EAS.

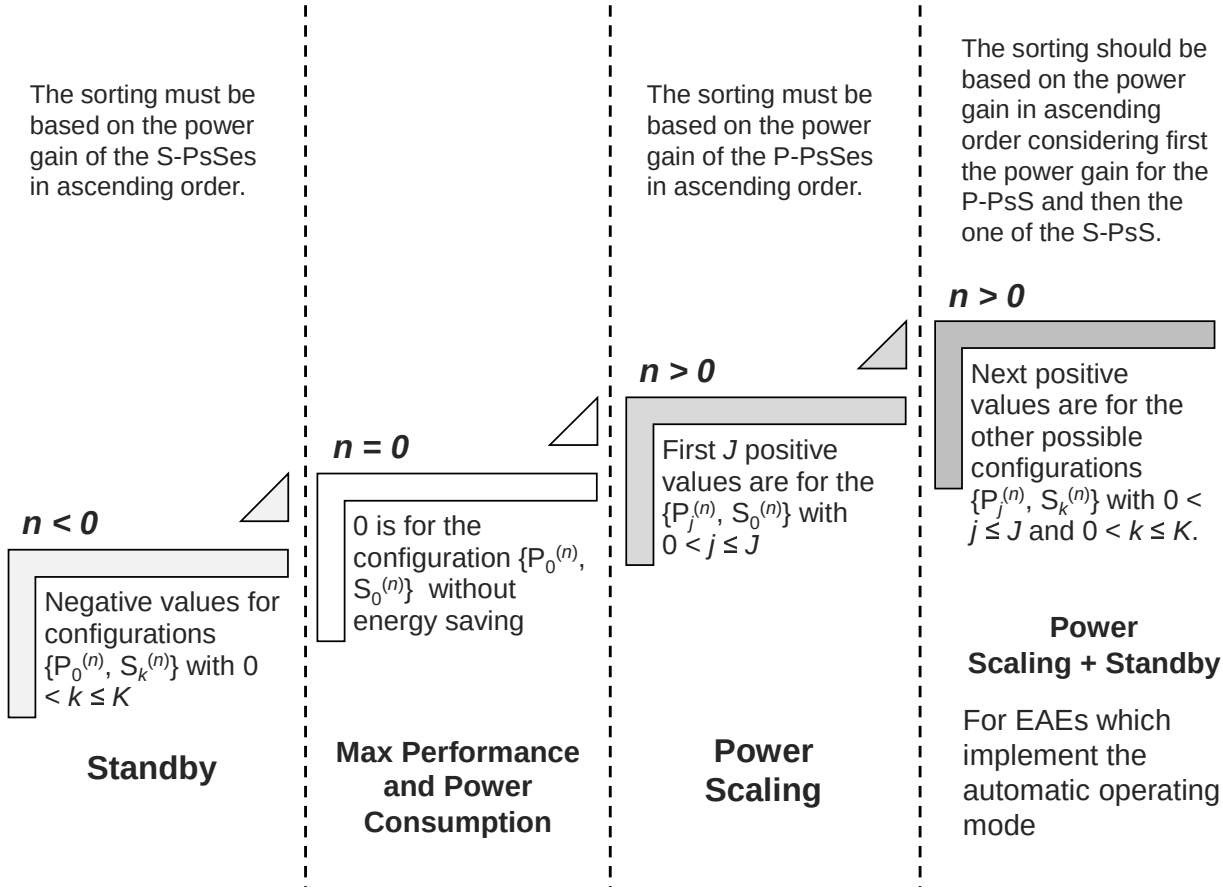

Figure 2: EAS identification mapping

To simplify the EAS setting and management, Figure 2 shows the mapping of the number *n* identifying a specific EAS that shall be chosen as follows:

- negative values of *n* shall regard the *K* standby configurations ($\{P_0^{(n)}, S_k^{(n)}\}$ with $0 < k \leq K$). The sorting shall be based on the power gain in ascending order;
- $n = 0$ corresponds to the pair with the highest power consumption (and hence the best performance) among the ones allowed by the EAE ($\{P_0^{(n)}, S_0^{(n)}\}$ configuration);
- the first *J* positive values of *n* shall regard the power scaling configurations $\{P_j^{(n)}, S_0^{(n)}\}$ with $0 < j \leq J$. The sorting shall be based on the power gain in ascending order;
- the next positive values of *n* refer to the other possible configurations $\{P_j^{(n)}, S_k^{(n)}\}$ with $0 < j \leq J$ and $0 < k \leq K$. The present document does not define all the possible configurations. Instead, the LCPs should choose which appropriate energy configurations have to be exposed. The sorting should be based on the power gain in ascending order considering first the power gain for the P-PsS and then the one of the S-PsS. Another possible way to sort can be by considering the average of the power gains of all pairs of P-PsSes and S-PsSes.

As shown in Figure 3, the total number of available PsSes is $K + J + 2$. The maximum number of pairs of PsSes (and hence of EASes) is $K + 1 + J + J \times K = (J + 1) \times (K + 1)$.

Finally, in order to make the rest of this clause more readable, we omit the index $^{(n)}$ of the EAS. Thus, we refer to the *j*-th P-PsS or to the *k*-th S-PsS with P_j and S_k, respectively.

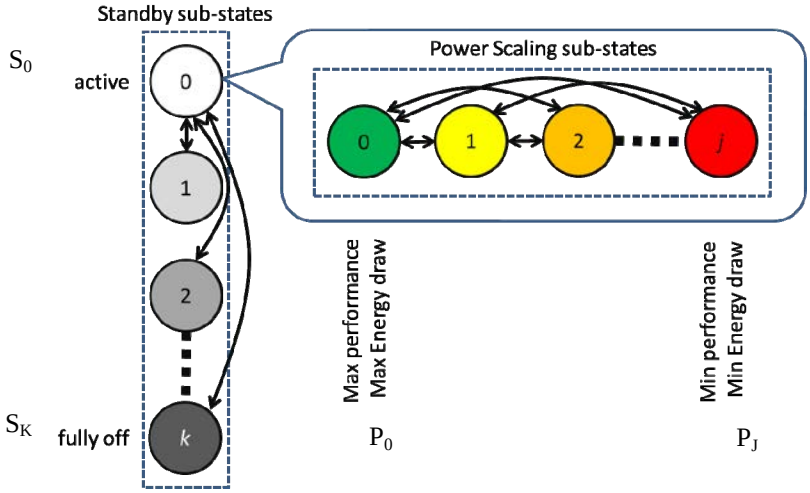

Figure 3: Organization of the Power Scaling and Standby PsSes composing EASes

An EAS, and consequently the related PsS pair, shall be accompanied by a set of attributes divided into three main categories:

- **power consumption parameters**: useful to estimate the global energy absorption of the entity when working with the selected state;
- **network performance indexes:** needed to verify if the current energy state meets the QoS requirements to serve the expected offered-load;
- **state's transition features**: allowing the control processes to evaluate the set of states reachable from the selected ones, giving information regarding transition delays and/or possible "*service interruption*" periods of the entity.

The power consumption of each P-PsSes and S-PsSes are expressed as the ratio with respect to the maximum power draw of the EAE. Then, these values are expressed as (power consumption [W] / max power draw [W]) $\times 2^{16}$.

The *autonomic behaviour* attributes aim at describing the performance of the autonomic energy-aware behaviour of the entity. This is done by describing the performance and the consumption of the entity for a number of offered load levels. The "*Autonomic Power Scaling Profile Curves"* parameter shall be a list where each node shall be composed by three values, indicating the offered load, the maximum consumption and the maximum packet service time at that load. The load shall be expressed as (current load [bps or pps] / max throughput of the EAE [bps or pps]) × 2^{16}. Instead, the maximum value shall be expressed as (max power consumption at the considered load [W] / max power consumption of the EAE) × 2^{16}. Finally, the maximum packet service time at the considered load shall be expressed in seconds.

The LCPs of medium/low levels are responsible to map these profile curves to the appropriate PsSes (both P-PsSes and S-PsSes) providing a configuration that does not exceed the constraints defined by the specific curve profile.

Table 9: EAS attributes

Parameter	Data Type	Optionality	Default Value	Value if not available	Description
n	`INTEGER`	Mandatory			EAS Identification number. In the GAL API the *eas_id* field uniquely identifies the EAE referring to this parameter.
P-PsS attributes	See Table 10.				
S-PsS attributes	See Table 11.				
Autonomic behaviour attributes					
Autonomic Power Scaling Profile Curves	`LIST <Measurement, Measurement, Measurement>`	Optional	NULL	NULL	
Autonomic Power Scaling Service Interruption	`Measurement` {*unit*="s"}	Optional	NULL	NULL	Maximum service interruption time that can be incurred when the autonomic power scaling mechanism switches the EAE's capacity.

6.1 Power Scaling Primitive sub-States attributes

The attributes of the Power Scaling Primitive sub-States (P-PsSes) are reported in Table 9. They are meant to represent AR primitives (if present on the EAE), and their operational behaviour (i.e., autonomic and non-autonomic).

In more detail, such attributes have been divided into three main categories, namely:

- *general*;
- *network performance*;
- *state transition*.

The *genera*l and *network performance* attributes are mandatory and are meant to provide a snapshot on the operating behaviour in the considered P-PsS. To this purpose, network performance and maximum power consumption bounds are defined, and try to summarize the aggregated behaviour of the entity without explicitly considering AR primitives and relative operational modes.

The third attribute category (*state transition*) gives indications about which other P-PsSes the entity can move to from the current one, and the related possible performance decays.

Only *general* and *network performance* attributes have been defined as mandatory, while the attributes of other categories shall be optional. This is mainly due to the possibility that the EAE does not include some of the primitives, whose characterizations are done in the third category of the attributes. On the contrary, all EAE elements shall be characterized at least by the *general* and *network performance* attributes.

Table 10: P-PsS attributes

Parameter	Data Type	Unit	Optionality	Default Value	Value if not available	Description
Minimum Power Gain	`Measurement`	Ratio	Mandatory			The minimum power with respect to the maximum power draw of the EAE.
Network Performance attributes						
Maximum Packet Throughput	`Measurement`	pps	Mandatory			The maximum throughput provided by the EAE when working in this EAS (i.e. in a specific P_j).
Maximum Bit Throughput	`Measurement`	Bps	Mandatory			The maximum throughput provided by the EAE when working in this EAS (i.e. in a specific P_j).
State transition attributes						
PsS Transition Times	`LIST <Measurement>`	s	Optional	NULL	NULL	List containing the maximum time to move from the current PsS to another one.
PsS Transition Service Interruption Times	`LIST <Measurement>`	s	Optional	NULL	NULL	List containing the maximum service interruption time that can be incurred while moving from the current P-PsS to another one.

6.2 Standby Primitive sub-States attributes

S-PsSes are represented by means of five attributes, as shown in Table 11. They are meant to represent LPI and standby primitives (if present on the EAE).

The "*Maximum Wake-Up Time*" attribute shall represent the maximum time required to move from the current S-PsS to the active sub-state ($k = 0$). On the contrary, the "*Maximum Sleeping Time*" attribute indicates the maximum time required to move from the active sub-state ($k = 0$) to the current S-PsS.

The "*Wake-up Triggers*" attribute represents the events that trigger the wake-up of the EAE. The possible values are: GAL, WoL, WoP, and internal. With the GAL value the wake-up is performed by a GAL command (e.g. the indication of the new current EAS). The WoL (Wake on LAN) and WoP (Wake on Packet) option allow an EAE to be awakened by a network message (known as "magic packet") or after the arrival of a packet, respectively. Instead, the last option "internal" refers to the possibility that an EAE can be awaken when entering the active state. For example, entities that provide the automatic operating mode should have this attribute set with the "internal" value.

Finally, the "*Activation Timer*" indicates the number of seconds after which an idle EAE with automatic operating mode transition enters standby modes in an autonomic way. Instead, for an EAE with manual operating mode transition this value shall be ignored.

Table 11: S-PsS attributes

Parameter	Data Type	Unit	Optionality	Default Value	Value if not available	Description
Power Gain	`Measurement`	ratio	Mandatory		NULL	The power with respect to the maximum power draw of the EAE.
State transition attributes						
Maximum Wake-Up Time	`Measurement`	s	Optional	NULL	NULL	The maximum time that is needed to wake-up from S_k to S_0.
Maximum Sleeping Time	`Measurement`	s	Optional	NULL	NULL	The maximum time that is needed to go to S-PsS S_k from active S_0.
Other attributes						
Wake-up Triggers	`ENUM {GAL, WoL, WoP, internal}`		Optional	GAL	Always available	It represents which triggers can be used to wake-up the EAE.
Activation Timer	`Measurement`	s	Optional	`Measurement` {*value*=0, *unit*="s"}	NULL	Time of inactivity before the EAE enters the specific S_k in an autonomic way. This attribute shall be ignored for EAEs with manual operating mode transition.

Annex A (informative): GAL Hierarchical Structure

The GAL architecture is designed as a modular and easily extendable Software (SW) framework, providing interface capabilities towards heterogeneous HW, as well as multiple hierarchical interfaces towards control processes, in order to provide and to set energy configurations at various detail levels of the internal architecture.

The GAL architecture is designed as an optional hierarchical structure, which permits to better exploit the functionality provided by this framework. For example, Figure A.1 shows the case of a multi-chassis network device including different HW components with energy-aware capabilities, which may be managed by control plane processes. Such device can be represented at various levels: single HW component and/or physical link levels, line-card, chassis and node levels.

The GAL allows the interaction between the power management usually realized inside HW components, and network control processes (e.g. routing and traffic engineering signalling protocols, etc.) run at the device level. When new device configurations are decided by such control processes, the GAL translates them into specific settings of components at underlying levels. For example, if the network-wide control decides to scale the speed of a link, it has to pass this command to the chassis, and then to the line-card hosting the link. The line-card can scale the speed of the physical link, as well as the speed of the related HW (which can be composed of multiple energy-aware components). Thus, a single configuration decided at the device level may impact on multiple underlying components.

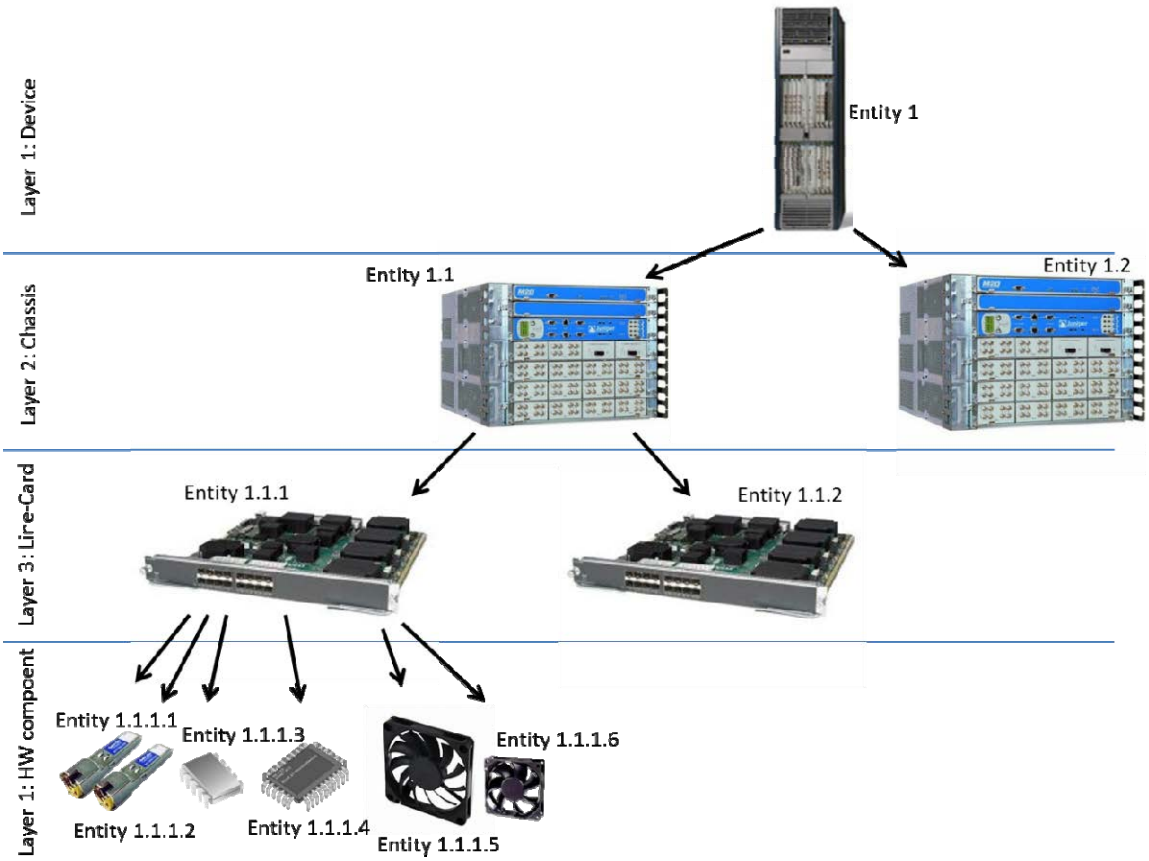

Figure A.1: The hierarchical architecture of a multi-chassis network device

Given the fact that power management primitives reside at the lowest levels of the hierarchy, they should require specific LCPs to directly manage them to achieve the desired operating behaviour. For this reason, the GAL is used for interfacing such lowest level EAEs with some control-plane processes implementing HW-specific optimization strategies.

Then, at intermediated levels (e.g. at line-card, or at chassis ones), new LCPs (one for each entity at that level) are needed to orchestrate the settings and the operating behaviours of underlying EAEs, and to expose a synthetic and aggregate set of operating characteristics and available configurations (of the line-card, or of the chassis) to higher levels.

This process terminates at the device level, where the highest LCP orchestrates the high-level configuration of the device, and needs to expose a simplified view of it to network signalling protocols (network-wide and OAM control applications).

The approach pursued here consists somehow of realizing a chain of LCPs, which control from single energy-aware HW components to the entire device. LCPs at different levels need to interoperate, and the highest level LCP needs to interact with processes realizing network signalling protocols.

The result of such hierarchical approach consists of a tree, where root nodes are LCPs and control applications, and leaf nodes correspond to HW elements. The interface among the tree nodes is realized by means of the GAL. Leaf nodes may reside at different levels of the hierarchy for two main reasons:

- Some HW EAE may be accessible at higher hierarchical levels (e.g. fans in a chassis).
- Some manufacturers may prefer not to expose the internal organization and the subcomponents of a "composite" part of the device (e.g. line-card, chassis, etc.).

In the latter case, the "composite" part of the device will be treated as a single HW EAE.

Annex B (informative): GSI Implementation

This clause describes the implementation workflow of the GAL. Note, however, that it is not intended to constrain the internal architecture of any conformant implementation.

B.1 Workflow

To generalize the GAL workflow of managing and monitoring the power configuration of hardware elements in a network device, let us consider a simple example of EAS configuration through the GAL, with the steps divided into five phases:

1) In the first phase, the NCP (e.g. routing protocol extended to consider energy-aware metrics) acquires the current EAS and the available EASes (with their associated attributes) from all network nodes.

2) In the second phase, upon the acquisition of all the available EASes and their associated attributes, on the basis of available traffic and network performance constraints, a centralized or distributed algorithm computes a new network configuration, which will include updates of the routing/switching tables and of configuration parameters in each node.

3) In the third phase, the local instance of the routing protocol updates routing/switching table entries and interacts with the device-level LCP to assure that the LCP works under the required configuration parameters, such as the set of EAS candidates, decision-making interval, and wake-up time when working in smart standby mode.

4) In the fourth phase, the device-level LCP searches which sub-entities are involved, and forwards the incoming request towards lower-level entities; this phase iterates by medium-level LCPs until it reaches the bottom-level LCP, which can directly access data-plane hardware components by means of the CLI.

5) In the fifth phase, the bottom-level LCPs dynamically configure the EASes of each component.

From this procedure, we can see that control-plane processes finally reach the entity of device level for acquiring EAS information, and provisioning or monitoring the state of each logical entity of the device. Therefore, each device-level entity should provide the User Interface for implementing the command set mentioned above. The main workflow of managing and monitoring the power configuration can be generalized as in Figure B.1.

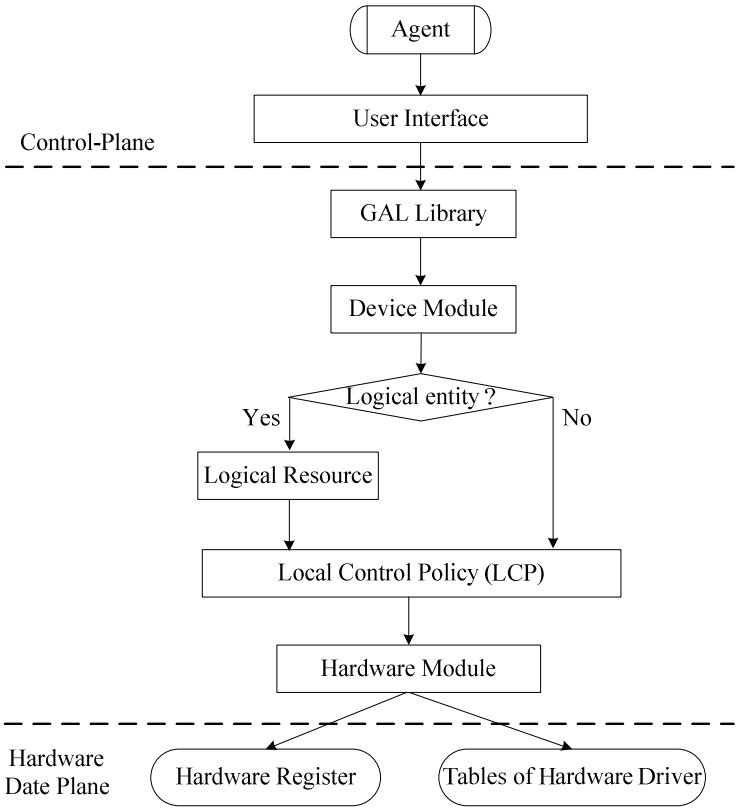

Figure B.1: Workflow of GAL Implementation

Summarizing, according to Figure B.1, the steps are:

- the control-plane process generates an appropriate control-message, which includes a protocol data unit (PDU) with the command and relative parameters for power management;
- the device-level entity has an Agent to receive control-messages. The agent performs a rudimentary parse of the incoming control-messages, and it passes the PDU to the User Interface which implements the desired functionality;
- the user Interface calls the corresponding functionality from the GAL Library;
- the corresponding functionality in the GAL Library forwards the request to the specific device module;
- if the command can reach the logical entity, the device module calls the corresponding functionality in logical resources, and the logical resource functionality passes the request to the Local Control Policy (LCP). Otherwise, the device module directly calls the LCP;
- the corresponding functionality of LCP interprets the high level states and sets the correspondent low level states to the hardware module;
- the module finally reaches the hardware component by accessing the corresponding registers or the table driver.

B.2 State Space

In order to realize the functions of these commands, each logical entity should use two variables to restore state information: *provisioned_state* and *committed_state*. The state of a logical entity is the combination of these two variables, and will change after different GAL commands.

Figure B.2 shows the logical entity state transition diagram, which describes the different possible states and the transitions between states (which are the consequence of the use of such commands). During the initialization, the *provisioned_state* and *committed_state* of each logical entity can be set with the *DEFAULT_STATE* value. The *provisioned_state* will be set with the *provisioned_state* and the *committed_state* will remain unchanged if the entity receives the **PROVISIONING** command. The **COMMIT** command will set the *committed_state* with the last provisioned *provisioned_state* while the *provisioned_state* remains unchanged. The **ROLLBACK** command will set the *provisioned_state* with the last committed *committed_state* while the *committed_state* remains unchanged. The **RELEASE** command will set the *provisioned_state* with *DEFAULT_STATE*, which has the same effect of the **PROVISIONING** command with parameter *DEFAULT_STATE*. The **DISCOVERY** and **MONITORING_STATE** commands can return either the current *committed_state* or *provisioned_state*, which depends on the parameter of these commands. Instead, the other MONITORING commands always return the information related to the current *committed_state* of the logical entity.

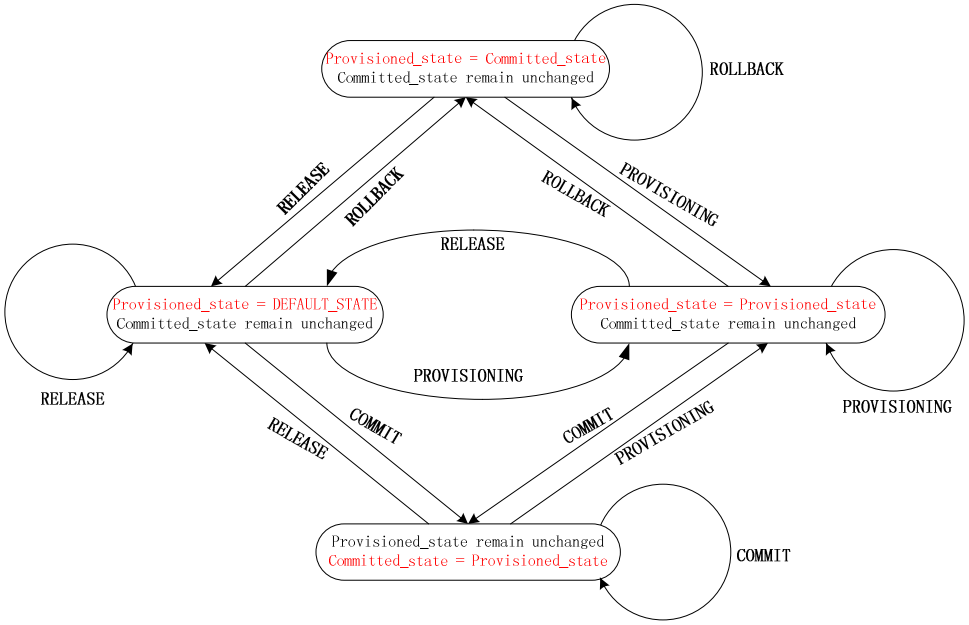

Figure B.2: Logical Entity State Transition Diagram

Annex C (informative): Other related standards

This annex underlines the main relationships of this proposal with relevant standardization works in the area of the energy-management for telecommunications infrastructures. Note, however, that this is an informative clause.

C.1 Energy Management (EMAN)

Beside the ACPI [i.5] specification, which is designed to establish industry common interfaces enabling robust operating system (OS)-directed motherboard device configuration and power management of general purpose PCs, the only current standardization action relevant to the GAL appears to be the IETF Energy Management (EMAN) [i.3], [i.4].

The EMAN addresses a different (larger) range of devices with respect to the GAL and it is oriented towards the collection of detailed data about the consumption and power states of the devices.

Nevertheless, given the different objectives, the GAL and the EMAN proposals can be implemented within the same network and even in the same device. In particular, the GAL might be applied "above" and/or "below" the EMAN to bind hardware and virtual/logical resources. In addition, given that the information model of EMAN allows the use of existing power states and describes a suggested set of states for control, administrators can use these states to create autonomic control systems for energy-aware applications by means of the GAL interface.

Annex D (informative): Trigger Attributes for EAS change

This annex introduces a new mechanism to better exploit the functionality provided by the Green Abstraction Layer (GAL). According to methods for control of power consumption modes of communications apparatus, a Network Management System (NMS) is provided. The network has nodes comprising communication apparatus capable of being operated in different power consumption modes, which provide different levels of performance. The network management system has a path computation apparatus configured to select paths for the traffic using the communications apparatus, based on information about traffic load in the network and on information about the power consumption modes of the communications apparatus.

According to the GAL framework, the multi-layer hierarchical interface allows the intercommunication among multiple local and network-wide control planes and the data-plane, the NCPs having bi-directional communication with the device's top-level LCP, meaning that the communication can be initiated by any of the two entities (i.e. NCPs from/to LCP).

As generic Sensor resource, the GAL makes use of *refresh_rate,* according to RFC 3433 [4] *entPhySensorValueUpdateRate*, implying a polling mechanism to retrieve the given information. This could be acceptable in small Networks, or more in general when it comes to retrieve (or monitor) non time critical information, such as power consumption, latency, etc. But when it comes to retrieve Actual Traffic Load in a large network, potentially consisting of hundreds or thousands of controlled systems, it is not expected that a centralized (or even distributed) polling-based global algorithm may react to fast changes in network demand in time constants of μs, ms or maybe even seconds (as an example, the most rapid gathering mechanism for Performance Counters is each 15 minutes).

A centralized approach carrying out a polling procedure to gather traffic load information where the NMS is cycling through many (e.g. hundreds or thousands) of nodes has at least the following disadvantages:

- overloading the Energy Aware NMS if polling is too frequent, while if it is too slow there is a risk of introducing an unacceptable delay in actuating a change; and
- a huge exchange of messages between the NMS and the (e.g. hundreds or thousands of) nodes is continuously needed with a polling solution, implying traffic load increase, power consumption increase, complexity and cost increase.

Indeed, in order to overcome these problems, it is necessary adding to the GAL a prompt way to transmit the bottom information of Traffic Load up to the NMS. In this way, the GAL can avoid these issues by using the communications equipment (node) to determine when an altered energy aware state is required, and by sending a request for an altered energy aware state at that time. The NMS is not required to monitor the traffic loads; instead, the NMS is informed of a change request, which is determined in the local controller. In other words, the communications apparatus makes a request for a different energy aware state when it detects that its traffic load has reached a given traffic threshold. The analysis to change to a different energy aware state is therefore off-loaded from the NMS.

Since the power mode controller can have more traffic information and more timely information, it is likely to be able to identify more quickly when communications apparatus can be put into a lower power mode, thus enabling better optimization of overall power consumption; the power consumption mode is controlled based on more up-to-date information about traffic loads, as well as without undue risk of loss of traffic due to delays in powering up again when there is more traffic. This can also help reduce the communications overhead between the controller and the communications apparatus. Moreover, it can help enable the controller to be scaled to work with many communications apparatus without too much processing and communications resources; it can make it easier for the network to tolerate many different types of communications apparatus.

The proposed solution needs to add a new parameter in the definition of the Standby Primitive sub-States:

- *"Sleeping Triggers*" - constituted by a set of flags. Each flag is meant to represent the events that may trigger the sleeping (or the deeper sleeping, i.e. $S_k \rightarrow S_{k+1}$) of the entity.

In addition, this method uses the "*Wake-up Triggers*" attribute in a different way with respect to the GAL one. With this attribute, we represent not only the events that may trigger the wake-up, but also the reduced sleeping (i.e. $S_k \leftarrow S_{k+1}$) of the entity.

Table D.1: Attributes to add to the S-PsS definition

Parameter	Data Type	Optionality	Description
Sleeping Triggers	UINT64	Optional	Set of flags that represent which triggers can be used to put the entity to sleep. Bit 0: GAL; Bit 1: sleep-on-no-packets; Bit 2: WoL. Other flags will be defined.

Similarly, it is necessary to add two new attributes to the P-PsS definition, too. In more details, these attributes are:

- "*Increasing Performance Triggers*" - constituted by a set of flags. Each flag is meant to represent the events that may trigger the increasing performance (i.e. $P_j \leftarrow P_{j+1}$) of the entity;
- *"Decreasing Performance Triggers*" - constituted by a set of flags. Each flag is meant to represent the events that may trigger the decreasing performance (i.e. $P_j \rightarrow P_{j+1}$) of the entity.

Table D.2: Attributes to add to the P-PsS definition

	Parameter	Data Type	Optionality	Description
Triggers Attributes	Increasing Performance	UINT64	Optional	Set of flags that represent which triggers can be used to increase the entity performance.
	Decreasing Performance	UINT64	Optional	Set of flags that represent which triggers can be used to decrease the entity performance.

D.1 Threshold based Actual Traffic Load Gathering

Figure D.1 shows a time chart of some additional steps in operating a network according to this embodiment. This is to illustrate the additional feature of changing power consumption mode (i.e. Energy Aware State) based on requests from the communication apparatus when it detects its traffic load has reached a given threshold. In Figure D.1 the left hand column shows actions of a communications apparatus and the right hand column shows actions of the power mode controller.

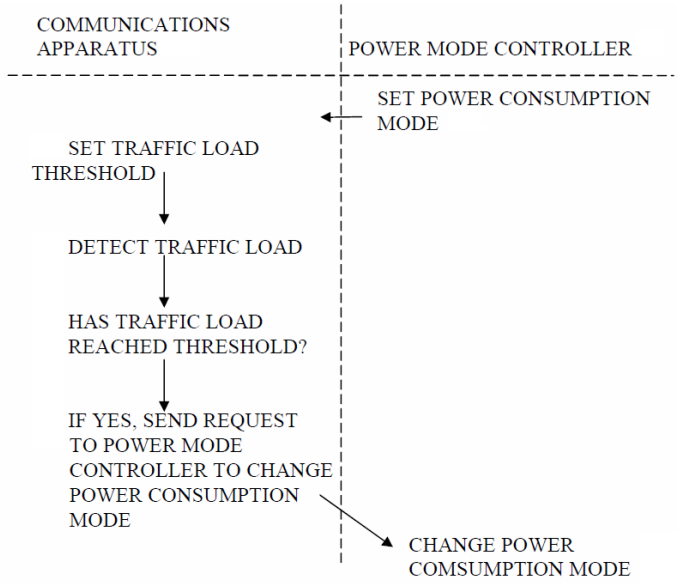

Figure D.1: Threshold based Traffic Load detection time flow

The power mode controller sets an initial power mode. The communications apparatus sets a traffic load threshold. The traffic load is detected at the communications apparatus, and compared to the threshold. If it has reached the threshold, then a request is sent to the power mode controller to change the power consumption mode. The power mode controller responds by changing the power consumption mode. In principle, the threshold can be an upper or a lower limit; in some cases there will be two thresholds to provide upper and lower limits. There can be different thresholds for each of the different power consumption modes.

D.2 Thresholds varying over time (sliding thresholds)

Figure D.2 shows a graph of traffic load and thresholds varying over time as time flows from left to right. The traffic load is shown represented as a bucket with a water filling level at three different time instants. At a first time instant the load is between an upper threshold (shown by a dark flag) and a lower threshold (shown by a light flag). The top of the bucket is the Energy Aware State ceiling indicating the maximum performance in terms of traffic load capacity. The upper threshold is a little below the ceiling so that there is some margin to allow time for the local controller or the Energy Aware NMS to take action. At a second time instant, the traffic load has increased sufficiently to cause a request to increase the capacity to be triggered. In response, the power consumption mode has been changed as shown by the increased ceiling. The upper and lower thresholds have also been changed to be higher than before (sliding thresholds).

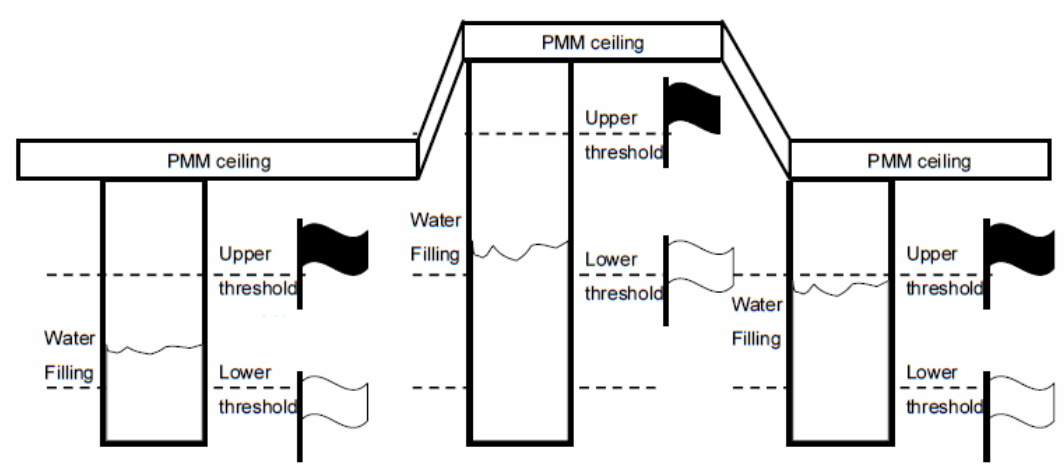

Figure D.2: Thresholds varying over time (sliding thresholds)

At the third instant the traffic load has decreased, so a request to decrease the power consumption mode has been sent and the mode has been changed, shown by the lower Energy Aware State ceiling, and the lowered upper and lower thresholds. According to threshold comparison outcome, the element can:

- Send an Urgent Request towards the EA NMS, for a higher performing Energy Aware State setting, as soon as Traffic Load is above the upper threshold (dark flag in Figure D.2).
- Send a Non-urgent Request towards the EA NMS, for a lower performing Energy Aware State setting, as soon as Traffic load is below the lower threshold (light flag in Figure D.2).
- Do not send any request (do not do anything), as long as Traffic Load is in between the two thresholds.

The EA NMS can promptly react to Urgent Requests by computing and providing the new (higher performing) Energy Aware State and/or new Traffic Routing configurations redistributing the traffic so to off-load the critical elements.

The EA NMS can react with a lower priority to Non-urgent Requests by computing and providing the new (lower performing) energy Aware State and/or new Traffic Routing configurations redistributing the traffic so to properly utilize capabilities. As long as the Traffic Load is in between the two thresholds (very likely for the vast majority of time), no actions should be taken.

D.3 Conclusions

A notable issue for any method aimed to dynamic adaptation as function of traffic load is an effective procedure to get the Actual Traffic Load; this is not necessarily an easy task, bearing in mind the following:

- An Energy Aware NMS could subtend a huge number of Network Elements.
- The Traffic Load get-procedure should be as prompt as possible, minimizing the time between the get and the correspondent Energy Aware State set, so nullifying the risk of QoS impacts by inconsistent setting due to undetected traffic load changes.
- A top-down cycling polling procedure risks overloading the Energy Aware NMS if too frequent, while if too slow it risks introducing unacceptable delay between Traffic Load gathering and consequent Energy Aware State setting (e.g. a gathering mechanism each 15 minutes, such as for Performances Counters, risks to be too slow to guarantee QoS or anyhow the optimal effectiveness of the energy saving mechanism; while it is still unable to guarantee not to overload the EA NMS when subtending a very large number of elements).

A solution to such issues is as follows:

- Each element can provide spontaneous indication (bottom-up) of the proper Traffic Load status just when strictly needed.
- For every EAS of each element two Traffic Load thresholds will be defined, upper and lower, such to discriminate if actual Traffic Load is in line with actual performance capability, or if it is getting dangerously close to max allowed capability, or if traffic load is too low, i.e. there is oversized capability.

Each element can compute autonomously the proper Traffic Load status by watching the local "packet counters" comparing the result with locally stored thresholds, selected as a function of the given (actual) Energy Aware State setting (the convenience to provide threshold levels update capability by the EA NMS, for instance during the Discovery Phase or overtime, can be evaluated).

By means of this method, the Energy Aware NMS is spared from the need to ask repeatedly for actual traffic load from any subtended element (the Energy Aware NMS can anyhow maintain the ability to request in a timely manner the exact percentage of the Traffic Load of a given element, for unexpected needs or for further refinement of the load distribution).

By detecting at the communications apparatus that the traffic load has reached a traffic threshold, the power mode controller can be relieved from regular polling and thus help reduce the communications overhead between the controller and the communications apparatus, and reduce the computational overhead at the controller. It can enable the communications apparatus to react more quickly without the latency and communications delays involved in polling. Moreover, it can help enable the controller to be scaled to work with many communications apparatus without too much processing and communications resources.

Annex E (informative): Bibliography

R. Bolla, R. Bruschi, F. Davoli, L. Di Gregorio, P. Donadio, L. Fialho, M. Collier, A. Lombardo, D. Reforgiato Recupero, and T. Szemethy: "The Green Abstraction Layer: A Standard Power-Management Interface for Next-Generation Network Devices," IEEE Internet Computing, vol.17, no. 2, pp. 82-86, March-April 2013.

R. Bolla, R. Bruschi, F. Davoli, P. Donadio, L. Fialho, M. Collier, A. Lombardo, D. Reforgiato, V. Riccobene, T. Szemethy: "A Northbound Interface for Power Management in Next Generation Network Devices," IEEE Communications Magazine, January 2014.

D. Reforgiato Recupero, A. Lombardo, F. Davoli, L. Fialho, M. Collier, P. Donadio, R. Bolla, and R. Bruschi: "Exporting data-plane energy-aware capabilities from network devices toward the control plane: The Green Abstraction Layer," 17th European Conference on Networks and Optical Communications (NOC), Vilanova i la Geltru, Spain, pp. 1-6, June 2012.

IETF Open Shortest Path First (OSPF) Working Group.

NOTE: http://www.ietf.org/html.charters/ospf-charter.html.

IETF RFC 2205: "Resource ReSerVation Protocol (RSVP)".

NOTE: http://tools.ietf.org/html/rfc2205.

IETF RFC 2210: "The Use of RSVP with IETF Integrated Services".

NOTE: http://tools.ietf.org/html/rfc2210.

IETF RFC 2211: "Specification of the Controlled-Load Network Element Service".

NOTE: http://tools.ietf.org/html/rfc2211.

IETF RFC 2212: "Specification of Guaranteed Quality of Service".

NOTE: http://tools.ietf.org/html/rfc2212.

ETSI ES 202 336-12: "Environmental Engineering; Monitoring and Control Interface for Infrastructure Equipment (Power Cooling and Building Environment Systems used in Telecommunication Networks), Part 12: Telecom/ICT equipment control and monitoring information model".

Patent application number PCT/EP 2012_069418: "Control of Power Consumption Modes of Communications Apparatus".

History

Document history		
V0.7.4	January 2014	Membership Approval Procedure MV 20140323: 2014-01-22 to 2014-03-24
V1.1.1	March 2014	Publication