Abstract—After the big Internet growing of the past decade, in the current decade a considerable effort is spent to reduce the Internet energy consumption. Actual Internet topologies have space to power off some links and devices in order to reduce energy consumed in off-peak periods still guaranteeing connectivity among terminals. In this paper we propose a methodology to identify less used links in the network in order to have the capability to switch off these network interfaces for energy saving purposes. We describe four different algorithms able to identify this set of links and we show the tradeoff between complexity, and the consequent execution time, and efficiency in powering off a great number of links. We find that, by using our solutions, it is possible to switch off a big percentage of links still keeping the network load under suitable thresholds and still guaranteeing topological characteristics of the resulting network topology.

Keywords—Energy consumption, Green Internet, Graph theory, Routing protocols.

I. INTRODUCTION

In the past decades a big effort has been dedicated to improve the Internet infrastructure by adding hardware and software systems that constitute essential components in its operation. Physical transmission lines along with routers constitute the skeleton if this infrastructure and they should provide network connectivity and reliability with satisfactory traffic support. However, in these years, its is emerging that on one side the power consumption of the Internet is about 0.4% of electricity consumption in broadband-enabled countries [1] and on the other side the Internet network topology has grown without a specific design thus giving rise to redundancies and resource wasting.

As a consequence, there is space in the current Internet topologies to power off some links and devices to reduce energy consumed in off-peak periods (e.g., during night) still guaranteeing connectivity among terminals [2], [3]. The position paper of Gupta et alii ([4]) stressed that sleeping appears to be an appropriate way to maximize energy conservation in the Internet. To this aim we propose a new approach based on the usage of links in network paths in order to define an algorithm to power off less-used links. We named this proposal ESOL, Energy Saving based on Occurrence of Links. We leverage routing information gathered during routing phases to measure occurrences of nodes and links in network paths. We then provide an effective method for reducing the number of active links by contemporary cutting a big slice of energy consumed to power on linecards of the network routers.

The main contributions of this work are:

- we identify two parameters, the occurrences of links and nodes, that can be simply extracted from the network topology by using the classical Dijkstra algorithm; these parameters are effectively used to select network interfaces (router linecards) that can be switched off;
- we propose four versions of ESOL that present different tradeoffs between complexity, and the consequent execution time, and efficiency in powering off a great number of links;
- we compare performance results of these four versions by applying them on topologies extracted by real IP Autonomous Systems;
- we compare performance results of our algorithms with another solutions proposed in the literature.

The paper is structured as follows. Section II introduces the adopted parameters and Section III describes the four versions of the ESOL algorithm. The analysis of the proposed solutions is presented in Section IV. Finally, Section V concludes the paper.

II. OCCURRENCE OF NODES AND LINKS

We model the Internet topology (i.e., the topology of an Autonomous System - AS) with a simple unidirectional graph $G(N, \mathcal{E})$, where $N$ is the set of routers and $\mathcal{E}$ is the set of unidirectional links connecting these routers. Let $N = |N|$ and $E = |\mathcal{E}|$ be the cardinalities of $N$ and $\mathcal{E}$, respectively. We indicate with $l_{ij}$ the link connecting router $i$ and router $j$ in the direction $i \rightarrow j$ ($i, j = 1, 2, \ldots, N$ and $i \neq j$). We suppose that an AS uses the Open Shortest Path First (OSPF) protocol to route IP packets. As well know this protocol gathers link state information from available routers and constructs a topology map of the network at every router. So we can assume that every router knows the network topology represented by $G$, and that it can compute, by means of Dijkstra algorithm, the set of all the shortest paths between every couple of nodes $s$ and $d$. We indicate with $(s, d)$ the shortest path between nodes $s$ and $d$ and with $(s, d) = (s, d) = G$ the set of all the shortest paths between every couple of nodes $s$ and $d$ in $G$. In this way it possible to compute how many times a router $i$ and a link $l_{ij}$ are present in these paths.
We define the occurrence of router $i$ in $\{(s,d)\}_{G}$ as:

$$O^G(i) = \sum_{s=1}^{N} \sum_{d=1}^{N} \sigma_i(s,d)$$

(1)

where

$$\sigma_i(s,d) = \begin{cases} 
1 & \text{if } i \in (s,d) \\
0 & \text{otherwise} 
\end{cases}$$

(2)

and $(s,d)$ represents the path from node $s$ to $d$.

As for nodes, we define the occurrence of link $l_{ij}$ in $\{(s,d)\}_{G}$ as:

$$O^G(l_{ij}) = \sum_{s=1}^{N} \sum_{d=1}^{N} \sigma_{l_{ij}}(s,d)$$

(3)

where

$$\sigma_{l_{ij}}(s,d) = \begin{cases} 
1 & \text{if } l_{ij} \in (s,d) \\
0 & \text{otherwise} 
\end{cases}$$

(4)

Roughly speaking, $O^G(i)$ and $O^G(l_{ij})$ measure the number of times router $i$ and link $l_{ij}$ are used in the network paths selected by Dijkstra for routing IP packets. It is quite intuitive that if a router or a link are used few times they are candidates for a complete (or partial) switch off, particularly where the traffic in the network is low.

### III. ESOL ALGORITHM

In this Section we describe the ESOL (Energy Saving based on Occurrence of Links) algorithm that exploits $O^G(i)$ and $O^G(l_{ij})$ defined in Section II. The idea is to order network links on the basis of their occurrence and switch off those less selected by OSPF (Dijkstra) for routing purposes: therefore, links with the smallest $O^G(l_{ij})$ are the first candidates to be powered off. The algorithm goes on until the removal of one or more unidirectional links causes the disconnection of the network. To check if the network is connected ESOL uses the parameter $O^G(i)$. In fact, a network is connected if it exists at least a path between every pair of nodes; consequently, each node $i$ must be connected via a path to all others $(N-1)$ nodes of the network and these $(N-1)$ nodes must be connected the node $i$. This means that for each node $i$, $O^G(i)$ has to be at least equal to $2 \cdot (N-1)$, since node $i$ has to appear $(N-1)$ times as source and $(N-1)$ times as destination: obviously $O^G(i)$ is greater than $2 \cdot (N-1)$ if node $i$ is used also as relay node in some network paths. To verify whether the network is connected, it is sufficient to verify the previous condition only for the node with the smallest occurrence, that is:

$$\min_{1 \leq i \leq N} \{O^G(i)\} \geq 2 \cdot (N-1)$$

(5)

We propose different versions of ESOL, that are described in the following sections: i) basic-ESOL ($b$-ESOL); ii) fast-ESOL ($f$-ESOL); iii) $(f+b)$-ESOL; iv) $(f \times 2)$-ESOL.

#### A. Basic-ESOL description

Pseudo-code 1 describes how the basic-ESOL ($b$-ESOL) works. The input is the initial network topology $G(N,E)$ and the outputs are the final topology $G^{\text{final}}(N,E-S)$ where $S$ is the set of links that $b$-ESOL allows to switch off and the number of performed iterations ($N_{\text{iter}}$). The unidirectional links $l_{ij}$ of the initial topology are sorted in decreasing order on the basis of their occurrences $O^G(l_{ij})$ in network paths $\{(s,d)\}_{G}$ (lines 2-6). Then $b$-ESOL selects from this ordered list links that can be powered down until the network remains connected (we use a binary variable netConn that is 1 if the network is connected and 0 otherwise). At each iteration, the algorithm creates the set $X$ that is composed by those links of $G^{N_{\text{iter}}}$ characterized by the smallest value of $O^G_{N_{\text{iter}}}(l_{ij})$ (line 12). To verify if these links can be switched off, $b$-ESOL evaluates if the inequality (5) holds in the network graph $G^{N_{\text{iter}}+1}$ obtained by deleting from the initial network graph links contained in $X$ and links already in $S$ (lines 13-18). If this is the case, links in $X$ can be switched off and added to the set $S$ (line 19) and the occurrences $O^G_{N_{\text{iter}}+1}(l_{ij})$ of the residual active links are computed (lines 20-22), otherwise the binary variable netConn is set to 0 and the algorithm stops (line 25).

The main drawback of $b$-ESOL is that it presents a slow convergence, since at each iteration it switches off a low number of links and consequently it needs a high number of iterations to derive the final network graph. For this reason, we propose a faster version, named $f$-ESOL, that aims at reducing the number of iterations.

#### B. Fast-ESOL description

Pseudo-code 2 describes the steps performed by fast-ESOL ($f$-ESOL). Input and outputs are the same of $b$-ESOL: also in this case unidirectional links $l_{ij}$ of the initial topology are sorted in decreasing order on the basis of their occurrences $O^G(l_{ij})$ in network paths $\{(s,d)\}_{G}$ (lines 2-6). We introduce $f$-ESOL to speed up the switching off process. The basic idea is to find the largest set $X$ of links that can be powered off in one iteration without disconnecting the network. This set is composed by those links whose occurrence is lower than a given threshold set to $N/K$ (line 12), where $2 \leq K \leq N$; consequently, the aim is to find the smallest value of $K$ that maintains the network connected. Therefore, at the first iteration, $K$ is set to 2 and the algorithm evaluates whether the condition expressed by Equation (5) holds in the network graph $G^{\text{reduced}}$ obtained by deleting from the initial network graph links contained in $X$ (lines 13-18). If this is the case, links in $X$ can be switched off and added to the set $S$ (line 19) and the procedure stops (line 20). Otherwise, if the network disconnects, $K$ is increased by one (line 22) and the procedure skips to the next iteration to verify if, by reducing the set of links to be switched off, the network remains connected. Notice that $f$-ESOL performs $N_{\text{iter}}-1$ iterations to find a suitable value of $K$ that allows to switch
off the largest set of links without disconnecting the network and it performs only one iteration to select the set of links to be powered down.

C.  \((f + b)\)-ESOL and \((f \times 2)\)-ESOL description

Compared with b-ESOL, f-ESOL is able to power off a high percentage of links in the set identified by b-ESOL in a significant lower number of iterations, however it does not reach b-ESOL performance in terms of number of switched off links. This happens since f-ESOL is not able to switch off links with occurrences greater or equal to \(N/K\), that instead could be switched off with b-ESOL. For this reason, we propose to combine these two methods by applying them in cascade, that is the network graph provided as output by f-ESOL is the input of b-ESOL. In this way we can switch off a big percentage of links in a fast manner with f-ESOL and then continue to slowly power off other links with b-ESOL. This solution is denoted as \((f + b)\)-ESOL. 

Another way to overcome the drawback of f-ESOL is to repeat in cascade f-ESOL two times: in this way the second f-ESOL can switch off another portion of links remained active after the first f-ESOL run. This solution is named \((f \times 2)\)-ESOL. Obviously this can be iterated multiple times but we empirically observed that two times are sufficient to have a behavior close to b-ESOL.

D. Algorithms complexity

All the proposed algorithms detect a subset of links that can be switched off. The complexity for their implementation is strictly related to the specific network topology: in particular, we suppose to operate in a Autonomous System where a link-state intra-AS routing protocol, such as OSPF, is adopted. The availability of the network topology in every network node (thanks to the OSPF database) makes it possible a distributed solution. The complexity of these algorithms is equal to \(O(NK \cdot \log N)\), since at each iteration the algorithms have to compute all the shortest paths between every source-destination couple. However this is a complexity already supported in all OSPF routers.

IV. Performance analysis

In order to evaluate ESOL performance (in all its four versions), we use real ISP topologies captured thanks to the Rocketfuel engine [5], namely EBONE \((N = 159\) and \(E = 614\)), EXODUS \((N = 244\) and \(E = 1080\)) and ABOVENET \((N = 366\) and \(E = 1932\)), in which each link is considered unidirectional. We measured the following performance parameters:

1) the percentage of links that each algorithm allows to switch off, denoted by \(\eta\);
2) the mean utilization of the remaining active links (i.e., on the links of network modeled by \(G^{fin} = (\mathcal{N}, \mathcal{E} - \mathcal{S})\)); this parameter is indicated with \(U\) and expressed in percentage.

---

**Algorithm 1 b-ESOL**

1. **Input:** a network graph \(G(N, E)\);
2. compute \(\{(s, d)\}^G\);
3. for each \(l_{ij} \in G\) do
4. compute \(G^2(l_{ij})\);
5. end for
6. sort in decreasing order the \(O^2(l_{ij})\) values;
7. \(S = \emptyset\);
8. \(netConn = 1\)/binary variable that indicates if the network is connected/*;
9. \(Niter = 0\);
10. \(G^{Niter} = G\);
11. while \(netConn = 0\) do
12. \(X = \{l_{ij} : G^{Niter}(l_{ij})\) is the smallest\};
13. \(G^{Niter+1} = (N, E - S - X)\);
14. compute \(\{(s, d)\}^{G^{Niter+1}}\);
15. for each \(l_{ij} \in G^{Niter+1}\) do
16. compute \(O^{G^{Niter+1}}(l_{ij})\);
17. end for
18. if \(\min_{1 \leq i \leq N} (O^{G^{Niter+1}}(i)) \geq 2 \cdot (N - 1)\) then
19. \(S = S \cup X\);
20. for each \(l_{ij} \in G^{Niter+1}\) do
21. compute \(O^{G^{Niter+1}}(l_{ij})\);
22. end for
23. \(Niter = Niter + 1\);
24. else
25. \(netConn = 0\);
26. end if
27. end while
28. **Outputs:** \(S\)/list of network’s links that can be switched off/*, \(G^{fin} = (N, E - S)\)/final network topology/* and \(Niter\).

---

**Algorithm 2 f-ESOL**

1. **Input:** a network graph \(G(N, E)\);
2. compute \(\{(s, d)\}^G\);
3. for each \(l_{ij} \in G\) do
4. compute \(O^G(l_{ij})\);
5. end for
6. sort in decreasing order the values \(O^G(l_{ij})\);
7. \(S = \emptyset\);
8. \(stop = 0\);
9. \(Niter = 0\);
10. \(K = 2\);
11. while \((stop=0)\) do
12. \(X = \{l_{ij} : O^G(l_{ij}) < N/K\};
13. \(G^{reduced} = (N, E - X)\);
14. compute \(\{(s, d)\}^{G^{reduced}}\);
15. for each \(l_{ij} \in G^{reduced}\) do
16. compute \(O^{G^{reduced}}(l_{ij})\);
17. end for
18. if \(\min_{1 \leq i \leq N} (O^{G^{reduced}}(i)) \geq 2 \cdot (N - 1)\) then
19. \(S = X\);
20. \(stop = 1\);
21. else
22. \(K = K + 1\);
23. end if
24. end while
25. \(Niter = K - 1\);
26. **Outputs:** \(S\)/list of network’s links that can be switched off/*, \(G^{fin} = (N, E - S)\)/final network topology/* and \(Niter\).
3) the energy that can be saved by applying each algorithm, denoted with \( \xi \) (W);
4) the increase of network paths lengths (expressed as number of hops) in \( G^{fin} \);
5) the number of iterations performed by each algorithm, denoted with \( N_{iter} \);
6) the mean number of powered off links per iteration, denoted with \( M \).

As for item 1), \( \eta = S/(E - E_{min}) \), where \( S = |\mathcal{S}| \) is the number of unidirectional links that can be switched off and \( E_{min} \) is the minimum number of unidirectional links that guarantees the network connectivity, that is shown to be \( E_{min} = 2 \cdot (N - 1) \).

As for the item 2) we generate a reference traffic matrix according to guidelines reported in [6]. In particular, we suppose that each router generates traffic towards any other router and the composition is: 40% of traffic is high bit-rate traffic, between 1Mbit/s and 80Mbit/s, and the remaining 60% is a low bit-rate traffic, up to 1Mbit/s. The capacity of each link, indicated with \( C_l \), is selected considering the availability of 2.5Gbit/s modules and imposing that the maximum link utilization, in the initial network, is lower than 25%. The mean utilization of the residual active links is computed as:

\[
U(\%) = \sum_{l=1}^{E-S} \frac{U_{l}^{fin}}{E - S} \cdot 100
\]

where \( U_{l}^{fin} \) is the utilization of the link \( l \) in the final network topology.

As for item 3), we assumed that each 2.5 Gbit/s module consumes \( \Xi = 140 \) W [7], therefore a link of capacity of \( C_l \) Gbit/sec consumes an amount of energy equal to \( (C_l/2.5) \cdot \Xi \) W. In addition we consider that a link cannot be completely powered down, because of redundancy and fast recovery reasons, so a switched off link is a link in standby mode and it consumes a certain amount of energy; in this case we suppose that a standby link consumes about 20% of an active link.

As for item 4) we evaluated, in the initial Internet topology \( G \), the number of hops of a path connecting each possible source router with each possible destination router. This is obtained by running Dijkstra as done in OSPF. Then we repeated the same analysis for the final topology \( G^{fin} \), measuring the percentage of paths whose length increases of a certain number of hops due to the removal of links included in \( \mathcal{S} \).

As for item 5), we can compare the speed of the described algorithms by comparing the number of iterations performed by each of these, since the operations executed in each iteration by the different algorithms are the same (computation of all the shortest paths between every couple of nodes \( s \) and \( d \) in \( G \), computation of occurrences of nodes and links in these paths, evaluation of the connectivity condition and power off of links in \( \mathcal{S} \)). Therefore, we can conclude that the lower is the number of performed iteration, the faster is the algorithm. Finally, as for item 6), \( M = S/N_{iter} \). The performance analysis can be subdivided in two parts:

- a comparison among different versions of ESOL;
- a comparison with the d-EAR algorithm [3].

A. Comparison among different versions of ESOL

Figure 1 and Figure 2(a) represent, in case of the three AS topologies, a comparison among the four different versions of ESOL, in terms of \( \eta \) and \( N_{iter} \) parameters, respectively. We can notice that, among the four different versions, \( b \)-ESOL and \( (f+b) \)-ESOL are the ones that switch off the greatest number of links: consequently the energy saving obtained in these two cases is higher with respect to \( f \)-ESOL and \( (f+2) \)-ESOL. The problem is that, \( b \)-ESOL powers down a low number of links in each iteration entailing a high convergence time. Actually, as shown in Figure 2(a), in case of \( b \)-ESOL the number of performed iterations is the highest one; instead, the combination of \( b \)-ESOL and \( f \)-ESOL is faster than pure \( b \)-ESOL, since a high percentage of links are switched off in a fast way by using \( f \)-ESOL and then \( b \)-ESOL continues, slowly, to powers off the rest of links. This means that between \( b \)-ESOL and \( (f+b) \)-ESOL, the second one works better, since it is faster and the number of switched off links is approximately the same. However, the \( N_{iter} \) parameter remains quite high even if \( (f+b) \)-ESOL is used. Therefore, if we are more interested in the algorithm speed, it is more convenient to use \( f \)-ESOL or \( (f \times 2) \)-ESOL. In fact, in these cases the number of iterations to be performed is significantly lower compared with \( b \)-ESOL and \( (f+b) \)-ESOL: in particular, the gain obtained in terms of \( N_{iter} \) is greater than the disadvantage in terms of not powered down links. Moreover, with \( (f \times 2) \)-ESOL it is possible to increase the number of powered down links with respect to pure \( f \)-ESOL with a negligible increase in the number of iterations performed.

Analogous considerations can be done on the speed of these four algorithms as highlighted by observing the \( M \) parameter (Figure 2(b)). \( f \)-ESOL and \( (f \times 2) \)-ESOL are both characterized by a mean number of switched off links per iteration significantly higher with respect to \( b \)-ESOL and \( (f+b) \)-ESOL.

Table I summarizes all the results, also showing that the mean utilization of the residual active links is quite low even if a big percentage of links are switched off. The utilization of the residual active links remains quite low, since the links that are switched off are the ones less used in all the shortest paths between every source-destination couple. This means that, when active, these links carry a low percentage of traffic and, if these are switched off, only this low percentage of traffic has to be redistributed on the remaining active links: consequently, the utilization’s increment on the residual active link is limited.
To better show this concept, in Figure 3 we plot, in case of EBONE, the utilization of all network links (identified by a link-id) for both the initial network $G$ and the final one $G^{fin}$ obtained after $b$-ESOL application. In the initial network, we impose a link capacity in order to have a maximum link utilization lower than 25%, as can be seen in Figure 3(a); obviously these utilizations increase when some links are powered down, but Figure 3(b) shows that the utilization of residual active links is above 50% for 1.42% of links, is between 50% and 40% for 2.85% of links, is between 40% and 30% for 9.69% of links and finally is lower than 30% for 86.04% of links. We derived these results in case of $b$-ESOL, since this is the version of ESOL that switches off the highest percentage of links. The results, in terms of links utilization, in case of $f$-ESOL and $(f \times 2)$-ESOL are expected to be better than these.

Finally, Figure 4(a) shows, for EBONE topology, the distribution of the path length increase when $b$-ESOL is used. It is possible to notice that about 80% of paths do not vary their length in terms of hops, about 20% of paths increase their length of only one hop and a negligible percentage of paths increase the length of more than two hops. Also in this case, results are reported in case of $b$-ESOL, since it is the ESOL version that obtains the worst performance in terms of increase of number of hops in the network paths; therefore, we can conclude that, even if a big slice of links are switched off, the achieved results are very satisfying.

### B. Comparison with d-EAR algorithm

In this Section we compare the different versions of ESOL with an energy saving method proposed in literature: distributed Energy-Aware Routing (d-EAR) [3].

d-EAR is a routing algorithm that is able to save energy performing the “exportation” mechanism. An oriented link is put to sleep forcing its source router, referred as the importer, to use the Shortest Path Tree (SPT) of a specific neighbor, referred as the exporter. The routers role, and so the links in energy save mode, is defined on the basis of the node-degree strategy: routers having the highest degree “export” their SPTs to all their neighbors. The d-EAR main feature is its OSPF compliance. Table II shows, for the
three considered topologies, results, in terms of $\eta$, $\xi$, and $U$, obtained by applying $d$-EAR. By comparing it with Table I we can notice that $f$-ESOL and $(f \times 2)$-ESOL (that among the four ESOL versions achieve the lowest value of $\eta$) switch off a higher percentage of links compared to $d$-EAR, being able to save a greater amount of energy. As for the mean utilization, $f$-ESOL and $(f \times 2)$-ESOL produce lower values of $U$ with respect to $d$-EAR even if they obtain slightly higher values of $\eta$; instead $b$-ESOL and $(f + b)$-ESOL have comparable values of $U$ with respect to $d$-EAR, but their $\eta$ is significantly higher. In Figure 5 it is plotted the network links utilization in the final topology when $d$-EAR is used, in case of EBONE: in this case some links have a final utilization that overcomes the 100%, and this means that the final network obtained with $d$-EAR is not effectively able to support the required traffic. Moreover, $b$-ESOL is able to fairly split the network traffic among the residual active links with respect to $d$-EAR, where the active links utilization is quite unbalanced.

We also show, by comparing Figure 4(b) with Figure 4(a), that $b$-ESOL obtains better results as for the paths lengths since in $d$-EAR the percentage of paths that maintain the initial number of hops is lower with respect to $b$-ESOL.

V. CONCLUSIONS

To go in the direction of using in efficient and sustainable way the Internet network infrastructure we propose a methodology to identify the less used network links in order to have the capability to switch (or put to sleep) these network interfaces for energy saving purposes. We then proposed four different algorithms to identify this set of links and we showed the tradeoff between complexity, and the consequent execution time, and efficiency in powering off a great number of links. $b$-ESOL is the most effective algorithm but it has a very slow convergence, on the opposite side $f$-ESOL is very fast but the percentage of links powered off is less than $b$-ESOL. A suitable combinations of $f$-ESOL and $b$-ESOL produces an increase in the switched off links. Finally a repetition of $f$-ESOL is a good approximation of $b$-ESOL with a very reduced execution time.

In the three real topologies analyzed in the paper it is possible to switch off a percentage of links that goes from 60% to 90% of total number of links, still guaranteeing network connectivity. Although this seems a too high percentage, we showed that the network capability to support the traffic is only slightly affected as also happens for the topological characteristics of the network paths in terms of their lengths. We then can conclude that the proposed solutions are effective in reducing the energy consumed by the network elements without compromising the vital behavior of the network. A correct use of this kind of methodology, e.g., during the underloaded periods of the network, could improve the Internet energy efficiency.

REFERENCES


