Distributed Algorithms for Green IP Networks

Aruna Prem Bianzino[a,b], Luca Chiaraviglio[a,c], Marco Mellia[a,c]
a) Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Torino, Italy
b) Institut TELECOM, TELECOM ParisTech, Paris, France
c) Dipartimento di Elettronica, Politecnico di Torino, Torino, Italy
Email: bianzino@telecom-paristech.fr, {chiaraviglio, mellia}@polito.it

***Abstract*—We propose a novel distributed approach to exploit sleep mode capabilities of links in an Internet Service Provider network. Differently from other works, neither a central controller, nor the knowledge of the current traffic matrix is assumed, favoring a major step towards making sleep mode enabled networks practical in the current Internet architecture. Our algorithms are able to automatically adapt the state of network links to the actual traffic in the network. Moreover, the required input parameters are intuitive and easy to set. Extensive simulations that consider a real network and traffic demand prove that our algorithms are able to follow the daily variation of traffic, reducing energy consumption up to 70% during off peak time, with little overheads and while guaranteeing Quality of Service constraints.**

I. Introduction

The power consumption of networking devices scales with the installed capacity rather than the current load [1]. Thus, for an Internet Service Provider (ISP) the network power consumption is practically constant, unrespectively to traffic fluctuations. However, actual traffic is subject to strong day/night oscillations [2]. Thus, many devices are underutilized during off-peak hours when traffic is low. This represents a clear opportunity for saving energy and several projects are aiming at enabling power adaptive technologies for network devices [3], [4], [5].

In this context, resource consolidation is a known paradigm for the reduction of the power consumption. It consists in forcing a sleep state for a carefully selected subset of network devices and use the rest to transport the offered traffic. This is possible without disrupting the Quality of Service (QoS) offered by the network infrastructure, since communication networks are designed for the *peak foreseen traffic request*, and with *redundancy* and *over-provisioning* in mind.

Starting from the seminal work [6], the problem of reducing power consumption in telecommunication networks has been widely studied in the last years (see [7] for an overview). In [8] we proposed efficient linear programming formulations and heuristics that can scale up to hundreds of nodes and thousands of links. In [9] authors evaluated the possibility of putting cables in sleep mode in bundled links; the possibility of applying sleep mode to nodes adopting a game-theoretic approach is proposed in [10]. Moreover, approaches ranging from traffic engineering [11], to routing protocols [12], and new architectures [13] have been investigated. These works tackle the minimization of power consumption by putting in sleep mode network elements, such as routers and links, showing that large savings are possible. However, to the best of our knowledge, all the previous solutions are completely centralized or require at least the presence of a central control node which computes the solution to be distributed. Moreover, they either assume the perfect knowledge of the traffic matrix, i.e., the amount of traffic sent from each node to every other node, at each given time [8], [9], [10], [11], or ignore the traffic flowing in the network [12]. These assumptions clearly limit their applicability and deployment in the current Internet architecture.

In this work, we follow a different approach: we propose a *fully distributed solution* that leverages the knowledge of *current link load* instead of current traffic matrix. We tackle the reduction of power consumption in ISP networks, focussing on links in particular, for which support for sleep states can be more easily introduced than for nodes. Note that links normally consume more than 40% of the total network power consumption in an ISP [8], due to the fact that long-haul connections are present. We aim at dynamically adapting the working state of links (i.e., transmitters/receivers and amplifiers in the physical layer), thus adapting the network capacity (and power consumption) to the current traffic load, while still guaranteeing to not overload resources.[1]

Our solution is designed to work at the IP layer. Nodes leverage on a traditional Link-State routing protocol, e.g., OSPF [14], properly augmented to exchange information about the link power state (i.e., *on*, or *sleep*) and current load, on the basis of which decisions are taken.

In [8] we first faced the consolidation problem supposing a perfect knowledge of the (i) traffic matrix and (ii) device state. Centralized solutions similar to the one proposed in this paper were investigated. The intuition of a distributed resource consolidation was then investigated in [15], where we have proposed an algorithm based on Q-learning techniques. Differently from [15], in this work we assume nodes take informed decisions about which is the best link to target instead of taking independent decisions as in [15]. Simulation results show that our solution outperforms [15], being able to considerably reduce the power consumption of ISP networks (up to 70% of link power saving during low traffic periods, about 20-30% on a long term average), with a negligible number of reconfigurations. At the same time, the proposed algorithms

This work has been supported by the FP7 Integrated Project "Econet - low Energy COnsumption NETworks" funded by the European Commission.

[1]Consolidation can be further extended to include higher level devices (e.g., routers linecards, backplanes, switching fabrics, and even entire routers), but it is not considered in this paper.

include intuitive and easy to set parameters, contrary to [15].

While results show that large savings are possible, modifications to current networks devices are required to fully support the sleep mode, which is out of scope of this paper. However, researchers from both industries and academia are working towards the integration of sleep mode capabilities into current network devices [3], [4], [5], [16].

II. Problem Formulation

The problem of finding the network configuration corresponding to the minimum power consumption, under specific traffic conditions and QoS constraints, can be formalized as an ILP problem [8]. We consider an IP transport network, that we represent as a directed graph $G = (V, E)$, where V is the set of vertices representing network nodes, ($N = |V|$), while E is the set of edges representing interconnection links ($L = |E|$). For every pair of vertices $\{s, d\} \in V$, at every given time t, there is an aggregate traffic request to be routed from s to d, denoted as $r^{sd}(t)$. The set of all the traffic requests, at a given instant of time, is denoted as Traffic Matrix, $TM(t)$. Traffic slowly changes with a daily periodicity, with a pace of the order of several minutes.[2] For the sake of simplicity, in the following we drop the dependency of time from all variables.

Given thus a time instant t, the objective of the optimization is the minimization of network power consumption:

$$\min \sum_{(i,j)\in E} x_{ij} P_{ij} \tag{1}$$

where $x_{ij} \in \{0, 1\}$ is a binary decision variable that takes the value of 0 if link (i, j) is in sleep state, 1 otherwise, and $P_{ij} > 0$ is the power consumption of link (i, j) when it is in on state.

Traffic requests are routed over the network, generating a level of flow $f_{ij}^{sd} \geq 0$ over any network link (i, j). f_{ij}^{sd} is related to r^{sd} by the following set of conservation constraints:

$$\sum_{i\in V} f_{ij}^{sd} - \sum_{i\in V} f_{ji}^{sd} = \begin{cases} -r^{sd} & \forall s, d \in V, j = s \\ r^{sd} & \forall s, d \in V, j = d \\ 0 & \forall s, d \in V, j \neq s, d \end{cases} \tag{2}$$

To preserve the QoS in the network, maximum link utilization is imposed, $\phi \in [0, 1]$ hereafter. This defines the following additional set of constraints:

$$\frac{1}{c_{ij}} \sum_{s,d\in V} f_{ij}^{sd} = l_{ij} \leq \phi \quad \forall (i, j) \in E \tag{3}$$

being $c_{ij} > 0$ and l_{ij} the capacity and total traffic load on link (i, j), respectively.

Finally, we consider that a link can not enter a sleep state if its load, on both directions, is non-null. This introduces the following set of constraints:[3]

$$2\, x_{ij} \geq l_{ij} + l_{ji} \quad \forall (i, j) \in E \tag{4}$$

[2]Since we are focussing on backbone networks aggregate flows are assumed.

[3]We assume to put into sleep mode both link (i, j) and (j, i) due to technological constraint. Extension to have independent decisions is straight forward.

which forces the variable x_{ij} to take the value 1 when traffic is greater than 0, and allows the value 0 only when $l_{ij} = l_{ji} = 0$.

Minimizing the total power consumption (1) while satisfying all the previously mentioned constraints results into a mixed integer linear program. It has been proved to be NP-hard, and it has been shown not to scale for big networks, unless network topology is organized in special structures [8]. Moreover, its solution requires (i) the knowledge of the $TM(t)$ at each time t, and (ii) a centralized unit controlling all the nodes. Both are technologically unfeasible hypothesis considering current and foreseen network architectures.

III. Algorithm Description

To overcome the limits of previous proposals, we devise a distributed solution which relies only on the knowledge of (i) the current topology configuration $\{x_{ij}\}$, and of (ii) the traffic load on the links $\{l_{ij}\}$. Periodic Link State Advertisements (LSA) are broadcasted in the network, describing the state of the links (i.e., configuration and load). LSAs are also used to broadcast eventual *critical states*, e.g., presence of unreachable destinations ($lastLSACriticalState = KO$ in Alg. 1 and Alg. 2). This guarantees all nodes have the same knowledge of network status, and can take consistent decisions. Finally, link power consumption can also be shared by means of LSAs.

Distributed choices regarding the power state of links are made. All nodes run the same algorithm to find the link to target. Two cases are possible, based on the critical state information carried by the last received LSA:

Last LSA critical state OK – If the network is in a normal working state, i.e., last LSA confirmed constraints (2) and (3) are not violated (line 1 of Alg. 1), one link is selected to be possibly switched off ($(i, j)^*$ in Alg. 1). An unambiguous policy must be defined to select which link is target of the sleep attempt, on the basis of the local knowledge available to all the nodes. Possible choices may be, but not limited to, (i) selecting the least loaded link (Distributed Least Flow - DLF - hereafter), a choice that would have the lowest possible impact on current traffic routing, or (ii) selecting the most power hungry link (Distributed Most Power - DMP - hereafter), a choice that would have the highest possible impact on the energy saving. We suppose that a tie-breaking rule is defined as well, e.g., using lexicographical order.

Each node maintains three FIFO queues to store the last links that (i) entered into sleep mode but no LSA confirmed yet that constraints are not violated – `to_be_verified` list; (ii) are in sleep mode and caused no constraint violations – `sleepLinks` list; (iii) caused a violation and thus should not enter sleep mode anymore – `tabu` list. `tabu` list has a maximum length of `maxLength` links.

Being $E^* = \{(i, j) | x_{(i,j)\in E} = 1 \wedge (i, j) \notin \texttt{tabu}\}$, the link selection policies may be formalized as:

$$\text{DLF}: \qquad (i, j)^* = \arg\min_{(i,j)\in E^*} \{l_{ij}\} \tag{5}$$

$$\text{DMP}: \qquad (i, j)^* = \arg\max_{(i,j)\in E^*} \{P_{ij}\} \tag{6}$$

Before putting link $(i, j)^*$ into sleep mode, all nodes check if the network would still be connected after its removal (line 2 of Alg. 1). This operation will be referred to as *connectivity*

check, hereafter. The check is performed through a simple graph exploration algorithm, like a Breadth-First Search. If the connectivity check fails, then all nodes append $(i,j)^*$ to the `tabu` list and no further action is taken.

If the *connectivity check* is positive, $(i,j)^*$ can enter into sleep state. Nodes i and j take care of this by means of some signaling protocol if required, and insert $(i,j)^*$ in the `to_be_verified` list. Finally, they broadcast a new LSA to share the new state $x_{(i,j)^*}$ (lines 3-4 of Alg. 1).

Nodes i and j then wait for the first LSA after a sleep decision. If it reports constraint violations (unreachable node or link overload), they quickly undo the last move by popping all links $(i,j)^*$ from the `to_be_verified` list and inserting them into the `tabu` list (lines 3 to 7 of Alg. 2). Otherwise, if the LSA does not advertise any problem, elements from the `to_be_verified` list are moved to the `sleepLinks` list (lines 8-9 of Alg. 2).

Last LSA critical state KO – If the last LSA before a choice is reporting any constraint violation, nodes react by bringing back to operational state some link which was put into sleep state (lines 9 to 12 of Alg. 1). Also the choice of the link to be switched on must be unambiguous with respect to the distributed knowledge. Possible criteria may be, but not limited to, selecting (i) the last link entering a sleep state (*LastSleep* hereafter), or (ii) the closer sleeping link to the congestion point (*Distance* hereafter). The *LastSleep* criterion is based on the intuition that a recently made change is more likely responsible for the current congestion state. On the other hand, the *Distance* criterion is based on the intuition that the link which is closer to the congested point may more likely help in draining the extra traffic flow and relieve congestion. Distance between a couple of links is defined as the number of nodes on the shortest path between the nodes responsible for such links, e.g., the nodes with lowest ID.

The nodes responsible for the selected link switch it on. This mechanism allows the algorithm to react to traffic surges, and to link failures that have to be recovered by turning on other resources.

Implementation Issues - Our solution requires nodes to run a link-state routing algorithm, through which eventual link overload occurrences are signaled to all the nodes in the network. This allows nodes to timely signal and quickly react to eventual network congestions. Opaque LSAs supported by OSPF [17] may allow to easily carry the additional information in practical implementations, without any change to the link-state protocol. Node disconnections from the rest of the network are prevented through the connectivity check mechanism, which is run before trying to put a link in sleep state, and thus those should be very unlikely. In the rare case some node is going to be disconnected due to incongruent information, a recovery phase can be implemented using some signaling protocol on links. LSA timings can be set in accordance to OSPF specifications [14], so that on average each node has to process N LSA messages every Δ_{LSA}. As standard practice, overhead can be controlled by dividing the routing domain into different OSPF areas and running separate instances of the algorithm on each area. Considering the link sleep entering procedure, it can occur without any traffic losses. The link can indeed enter into sleep mode after all nodes routing tables have been properly changed, to allow a smooth traffic migration. Finally, loose synchronization is achieved among nodes by means of LSA messages, thus a strict synchronization is not required. Indeed, since the goal of the algorithm is to track the slow variation of traffic during the day, responsiveness to traffic changes is not critical.

```
Distributed Choice
Input: (i,j)*, lastLSACriticalState
1     if lastLSACriticalState == OK:
2       if connectivityCheck(x_(i,j)* = 0) == OK:
3         x_(i,j)* = 0
4         to_be_verified.append(x_(i,j)* = 0)
5       else
6         tabu.append((i,j)*)
7         if length(tabu) > maxLength:
8           removeOlder(tabu)
9     else:
10     ij = selectLink(sleepLinks)
11     x_ij = 1
12     sleepLinks.remove(ij)
```

Alg. 1: The pseudo-code of the *choice* event.

```
LSA receipt
Input: LSACriticalState
1   while length (to_be_verified) > 0:
2     ij = removeOlder(to_be_verified)
3     if LSACriticalState == KO:
4       tabu.append(ij)
5       if length(tabu) > maxLength:
6         removeOlder(tabu)
7       x_ij = 1
8     else:
9       sleepLinks.append(ij)
```

Alg. 2: The pseudo-code of the *LSA critical state reception* processing.

IV. Performance Evaluation

The algorithms have been implemented in a custom event-based simulator. Events correspond to traffic changes, LSA broadcasting events, and choice events.

LSAs are broadcasted every Δ_{LSA}. The time interval between two consecutive choices is a random variable, t_c which is uniformly distributed between Δ_{LSA} and Δ_c seconds. Indeed, LSA should be more frequent than choices (i.e., a choice, and its result, are notified before a new one takes place). The offered traffic is defined by $TM(t)$, which remains constant over intervals of length Δ_{TM}, and then changes to a new TM. Traffic is modeled as fluid and routed according to a minimum cost path algorithm. Link weights are given.

To provide a relevant evaluation of the described algorithm, we consider a benchmarking scenarios that we obtained from an actual nation-wide ISPs in Italy. The considered network is composed by 373 nodes and 718 bidirectional links. This topology is organized in different levels of nodes, among which some are pure transit nodes. We refer the reader to [8] for a detailed description of the network. For this scenario, link capacities and lengths are provided. Traffic is routed using link weights inversely proportional to the link capacities, a

TABLE I: Simulation scenario characteristics.

Parameter	Symbol	Value
Maximum Link Load	ϕ	50%
TM change interval	Δ_{TM}	48 min
Number of nodes	N	373
Number of links	L	718
Average link length	$E[m_{ij}]$	41 km
Average link capacity	$E[c_{ij}]$	6 Gb/s

common practice in ISP networks. Traffic matrix follows a day/night variation as reported in [8]. A summary of the main characteristics of the network is detailed in Tab. I.

Focussing on link power consumption, we adopt the same power model introduced in [8], and already used in [15], since it is representative for current actual devices. In particular, we consider network interfaces consuming P_{nic} = 50 W for each c_{ref} = 10 Gb/s of capacity. Electronic regenerators consume P_a = 1 kW for each c_{ref} = 10 Gb/s of capacity, with a regenerator that is required every m_a = 70 km of link length. Links whose capacity exceeds 10 Gb/s are formed considering a trunk of $k = \lceil \frac{c_{ij}}{c_{ref}} \rceil$ 10 Gb/s channels. The power consumption P_{ij} of link (i, j), of capacity c_{ij} and length m_{ij} results:

$$P_{ij} = \left\lceil \frac{c_{ij}}{c_{ref}} \right\rceil \left(\left\lfloor \frac{m_{ij}}{m_a} \right\rfloor P_a + 2P_{nic} \right) \quad (7)$$

While in this paper we focus on power consumption of links only, the algorithms can easily be extended to consider more fine grained models, e.g., considering the power consumption of linecards, switching fabrics and routers themselves. In the choice procedure, these power costs can be considered to select the best element worth trying entering sleep mode.

All simulations consider a one-week long period of time. This allows us to obtain average performance estimation, with negligible variations among different runs.

A. Time Evolution of the Topology Properties

Unless otherwise specified, we adopt the following set of parameters: Δ_c = 20 s, Δ_{LSA} = 10 s, `maxLength` = 70 links, which corresponds to nearly 10% of links, and maximum link utilization ϕ = 50%. Note that LSA timing matches current OSPF specifications [14].

Fig. 1a reports the power saving computed with respect to the scenario in which all links are on. DLF-Distance and DMP-LastOff algorithms are shown during the initial 5 days of system evolution using dotted blu and solid red lines, respectively. The dotted green line shows the power saving of the MP centralized heuristic, which has been proven in [8] to be the most effective one for this topology. The solid black curve reports the optimal solution computed solving the ILP problem of Sec. II for each TM[4]. Recall that the ILP solution allows single traffic requests to be split over multiple paths, which guarantees higher power savings. It has thus to be considered as an upper bound.

Several considerations hold. First, both DLF-Distance and DMP-LastOff are able to quickly follow traffic variation, and

[4]The solution has been obtained running CPLEX on a high performance cluster hosted in our Campus [18].

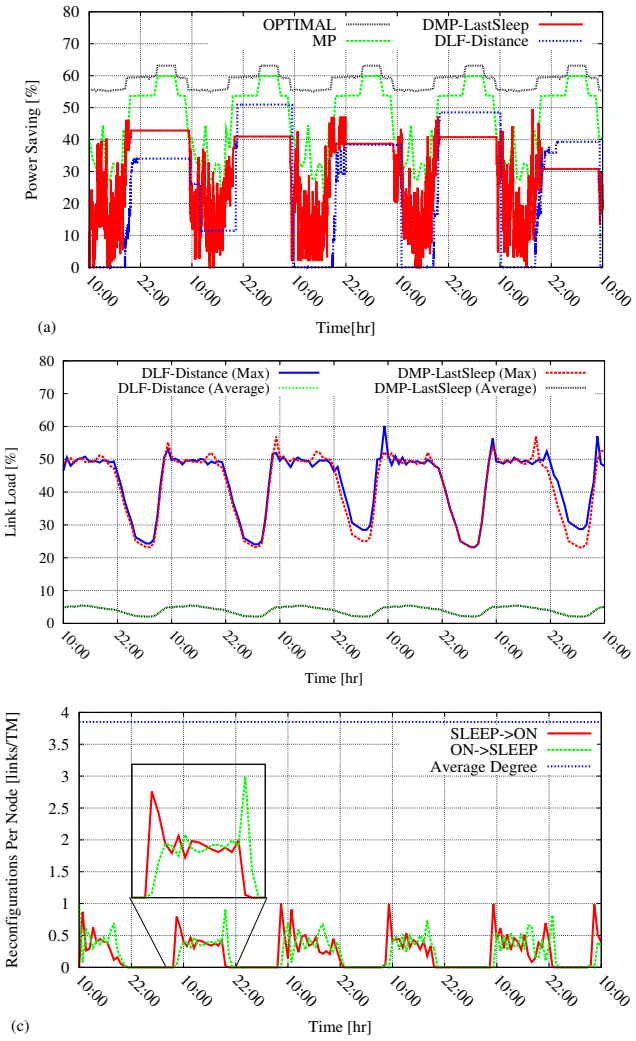

Fig. 1: (a) Power Saving, (b) Maximum and Average link load, (c) Average Number of Reconfigurations.

to achieve a good power saving. Since Δ_c is much smaller than the TM change frequency, transients are very quick. Both guarantee 30-50% of saving during night-time, which is comparable to the MP centralized heuristic, but without the perfect knowledge of the TM. Second, constant saving during night-time suggests that algorithms have converged to a solution which remains stable. Third, DMP-LastOff algorithm provides on average better performance than DLF-Distance in terms of power saving. Intuitively, power-hungry links are targeted by DMP which thus can achieve a better power saving even if putting into sleep mode a smaller number of links.

Finally, note that during the day, i.e., when more capacity is needed to meet traffic demand, both algorithms keep looking for possible links to enter sleep mode. DMP targets the most expensive links whose power cycling is reflected in the noisy power saving of Fig. 1a.

Fig. 1b details the average and the maximum link load over time. Considering the average load, we observe that its variation is limited during the day, suggesting that the algorithms efficiently match the current network capacity to the actual demand. Still, an over-provisioning of capacity

TABLE II: Algorithm Comparison, $\Delta_{LSA} = 10$, $\Delta_c = 20$

Algorithm	Saving [%]	Unacc. Choices [%]	ξ
OPTIMAL	58.56	–	–
MP [8]	46.28	–	–
GRiDA [15]	19.73	52	5.93e-4
DMP-Distance	32.35	20	3.23e-3
DMP-LastSleep	30.30	23	4.37e-3
DLF-Distance	25.45	17	1.63e-3
DLF-LastSleep	19.66	18	4.81e-4

is present since the average link load is smaller than 10%. Interestingly, during night-time, also the maximum link load is below 30%, i.e., far from the load threshold $\phi = 50\%$. This suggests that the connectivity constraint (2) is stricter than the maximum load constraint (3). During high traffic periods, (3) instead becomes predominant, and some link load actually gets close to 50%. Indeed, some violations are present, even if of short duration and of small intensity. We will better quantify violations in the next Section.

Finally, Fig. 1c reports the average number of SLEEP→ON (solid red line) and ON→SLEEP (dotted green line) changes per node per Δ_{TM}. DLF-Distance is considered, being results similar for other algorithms. Average node degree $\frac{2L}{N} = 3.85$ is plotted using dotted blu line as reference. During the night no choices occur confirming that the algorithm has converged to a stable solution limited by connectivity check. During early morning, the SLEEP→ON events are predominant and nodes switch back on some links to quickly react to traffic surge (see the inset). During the day, both events occur since the system continuously tries to adapt the network capacity to the actual traffic demand, but most changes are then undone due to temporary overload. Finally, during the evening the ON→SLEEP events become predominant due to traffic drop. Note that the number of reconfigurations per node is always limited to 1 per Δ_{TM} (i.e., 48 min), and much lower on average.

B. Average performance

We now compare the performance of different algorithms to assess which policy performs better. We consider energy saving, number of unaccepted choices, and network overload. Energy saving is evaluated as the integral of link power saving over a one week long time interval. Unaccepted choices account the percentage of sleep choices which are undone due to the immediate critical state indication by LSA. Percentage is computed with respect to the total number of sleep attempts. The network overload is defined as the fraction of traffic exceeding the load threshold ϕ with respect to the total carried traffic, i.e.:

$$\xi = \frac{\int_t \sum_{(i,j)\in E} \max(l_{ij}(t) - \phi, 0) dt}{\int_t \sum_{s,d\in V} r^{sd}(t) dt} \tag{8}$$

This is a relative indicator for the network congestion level, averaged over the simulation period, accounting for the number of load violations, their entity, and their duration. Note that we refer here to *violations* for link load overcoming the ϕ threshold, i.e., 50% of the link capacity. Link load never overcomes the full link capacity, i.e., 100%, in the considered scenarios. Let us consider, e.g., an average of 20 violation occurrences per hour, each one lasting 20 seconds on average, each one corresponding to a load of 5.5 Gbps over 10 Gbps links (i.e., $l = 55\%$). This will correspond on the considered network to an overload in the order of magnitude of e-3.

Tab. II reports results considering the distributed algorithms. For comparison, we include the optimal solution, MP [8] and GRiDA [15]. As the intuition already suggested from Fig. 1, higher energy savings are guaranteed by the DMP policies given that DLF policies are able to put in sleep mode links that do not consume much power. Note that up to 32.35% of energy saving can be reached for the considered network scenario, which is smaller than the centralized solutions but higher than [15].

The DMP algorithms are also more aggressive than the DLF policies in terms of sleep attempts, thus the percentage of unaccepted changes and network overload are higher for the formers. This is due to the fact that DMP targets energy hungry links, which are also the ones that carry lot of traffic being backbone links. Putting in sleep state one of these links results in a large amount of traffic to be re-routed over alternative paths. This causes a larger number of violations. Note that only less than 23% of choices results in a traffic violation.

To gauge how critical are those violations we focus our attention to the network overload ξ. Results confirm that violations are overall very small (see Fig. 1b) . This because the algorithms react to a critical state by immediately undoing the last power off attempt. Since LSA are frequently exchanged, the overload condition lasts no more than Δ_{LSA} in the worst case, i.e., few seconds.

Furthermore, comparing the LastSleep and Distance policies, we observe that the latter generally saves more power while showing quicker reaction to critical situation. This is due to the different reactions to traffic surges. Indeed, when a link is overloaded due to an increase of some r^{sd}, it is better to turn on some adjacent link, rather than the last link that has entered sleep state (being such link in any uncorrelated place in the network).

Finally, observe that all proposed algorithms outperform GRiDA. In particular, the number of unaccepted choices is much higher for GRiDA which is designed to turn on/sleep more than one link per choice, thus causing a large number of "wrong" decisions.

C. Sensitivity to Parameter Setting

We evaluate the impact of parameter choice on the performance. In particular, we consider the sensitivity to choice interval Δ_c, size of system memory `maxLength`, and LSA interval Δ_{LSA}. Parameters are varied one at a time, keeping the others set as reported in the previous section. Average results are computed over 7 days of simulations.

We start looking at the sensitivity of the algorithms to the time interval at which choices about links are made (Δ_c). Fig. 2 reports the variation in terms of network overload (ξ), and percentage of unaccepted choices, for increasing values of Δ_c. Interestingly, the percentage of unaccepted changes rapidly decreases as Δ_c increases, for all the algorithms.

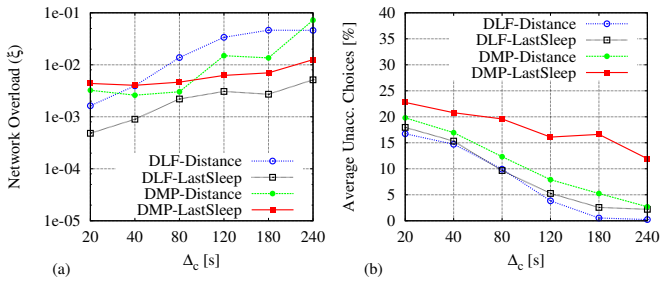

Fig. 2: Impact of Δ_c. (a) Network Overload, (b) Unaccepted choices.

TABLE III: Impact of `maxLength` on DLF-LastSleep (LS), and DLF-Distance (Di) algorithms.

maxLength [links]	Saving [%]		Unacc. Choices [%]		ξ	
	Di	LS	Di	LS	Di	LS
1	22.10	22.43	19	20	6.91e-4	1.29e-3
2	23.39	22.30	17	20	9.54e-4	1.13e-3
4	25.58	23.83	17	20	9.34e-4	1.23e-3
20	26.21	22.51	16	19	1.68e-3	8.48e-4
70	25.45	19.66	17	18	1.63e-3	4.81e-4

However, this is not beneficial for the network since the network overload steadily increases, suggesting that the system becomes slower in reacting to the changes of traffic. For example, with a choice made every two minutes on average, i.e., $\Delta_c = 240$ s, the average percentage of unaccepted choices is steadily below 5% for DLF-Distance, but the network overload is nearly two orders of magnitude higher than in the $\Delta_c = 20$ s case. Energy saving is 25% and 20% for $\Delta_c = 20$ s and $\Delta_c = 240$ s, respectively, confirming that the algorithm performance degrades for large values of Δ_c.

We then investigate the impact of the `tabu` size on the algorithm performance. Tab. III reports the metrics for different values of `maxLength`, obtained by running DLF-Distance and DLF-LastSleep on the considered scenario. Interestingly, the size of the `tabu` list on this scenario has a rather limited impact on performance. In particular, the percentage of unaccepted choices is decreasing as `maxLength` is increasing, suggesting that, as the system exploits more memory, the number of choices leading to negative LSA is decreased. However, it is also crucial not to have a too big memory to follow the traffic fluctuations. If the `tabu` list is too long, indeed, links are blacklisted for long periods of time during which they are kept on. Thus, the best savings are obtained as the length of the buffer is set to intermediate values.

Finally, we vary the LSA frequency, considering also the case when Δ_{LSA} is greater than Δ_c. Intuitively, a low LSA rate may deteriorate the algorithm performance since in this scenario node choices are based on a network status not constantly updated so that traffic changes can cause overload situations to which the system does not promptly react. Tab. IV reports the variations of the performance indicators for the considered network scenario with $\Delta_{LSA} \in [2s, 30s]$. Results consider the DLF-Distance. Interestingly, all metrics present just minor oscillations with respect to Δ_{LSA}, suggesting that the algorithm is robust even for high values of the parameter.

TABLE IV: Variation of Δ_{LSA}.

$\Delta_{LSA}[s]$	Saving [%]	Unacc. Choices [%]	ξ
2	24.22	18	8.84e-04
10	25.45	17	1.63e-03
20	22.53	17	1.13e-03
30	23.33	17	8.16e-04

V. Conclusions

We have presented a set of distributed algorithms to reduce power consumption in backbone networks. Results, obtained over realistic case studies, show that our algorithms achieve performance comparable to the optimal solutions and centralized heuristics, but without requiring the knowledge of the traffic matrix nor the presence of a central control node. We believe this makes the adoption of sleep mode policies practical in current IP networks.

As next step, we plan to integrate strategies that exploit sleep mode of other devices, e.g., routers linecards whose links are all in sleep mode, then routers switching fabrics whose linecards are all in sleep mode, etc.

References

[1] A. Adelin, P. Owezarski, and T. Gayraud, "On the Impact of Monitoring Router Energy Consumption for Greening the Internet," in *IEEE/ACM International Conference on Grid Computing (Grid 2010)*, (Bruxelles, Belgique), October 2010.
[2] What Europeans do at Night, "http://asert.arbornetworks.com/2009/08/what-europeans-do-at-night/," 2009.
[3] ECONET Project, funded by the European 7th Framework Programme, "http://www.econet-project.eu."
[4] GreenTouch Consortium, "http://www.greentouch.org/."
[5] D-Link Green Products, "http://www.dlinkgreen.com/greenproducts.asp."
[6] M. Gupta and S. Singh, "Greening of the Internet," in *ACM SIGCOMM 2003*, (Karlsrhue, Germany), August 2003.
[7] A. Bianzino, C. Chaudet, D. Rossi, and J. Rougier, "A Survey of Green Networking Research," *IEEE Communication Surveys and Tutorials*, no. 2, 2012.
[8] L. Chiaraviglio, M. Mellia, and F. Neri, "Minimizing ISP Network Energy Cost: Formulation and Solutions," *IEEE/ACM Transactions on Networking, to appear*, 2011. http://www.telematica.polito.it/chiaraviglio/papers/GreenTon.pdf.
[9] W. Fisher, M. Suchara, and J. Rexford, "Greening Backbone Networks: Reducing Energy Consumption by Shutting Off Cables in Bundled Links," in *1st ACM SIGCOMM Workshop on Green Networking*, (New Delhi, India), August 2010.
[10] A. Bianzino, C. Chaudet, S. Moretti, D. Rossi, and J. Rougier, "The Green-Game: Striking a Balance between QoS and Energy Saving," in *23rd International Teletraffic Congress (ITC 2011)*, (San Francisco, USA), September 2011.
[11] N. Vasić and D. Kostić, "Energy-aware traffic engineering," in *1st International Conference on Energy-Efficient Computing and Networking (e-Energy 2010)*, (Passau, Germany), April 2010.
[12] A. Cianfrani, V. Eramo, M. Listanti, M. Marazza, and E. Vittorini, "An Energy Saving Routing Algorithm for a Green OSPF Protocol," in *IEEE INFOCOM, 2010*, (San Diego, USA), March 2010.
[13] K. Ho and C. Cheung, "Green distributed routing protocol for sleep coordination in wired core networks," in *IEEE 6th International Conference on Networked Computing*, (Gyeongju, Korea (South)), May 2010.
[14] J. Moy, "OSPF Version 2." RFC 2328, April 1998.
[15] A. Bianzino, L. Chiaraviglio, and M. Mellia, "GRiDA: a Green Distributed Algorithm for Backbone Networks," in *2011 IEEE Online Green Communications Conference (GREENCOM 2011)*, September 2011. http://www.telematica.polito.it/chiaraviglio/papers/GRiDA.pdf.
[16] R. Bolla, R. Bruschi, A. Cianfrani, and M. Listanti, "Enabling backbone networks to sleep," *Network, IEEE*, vol. 25, no. 2, pp. 26–31, 2011.
[17] R. Coltun, "The OSPF Opaque LSA Option." RFC 2370, July 1998.
[18] POLITO HPC Initiative, "http://dauin-hpc.polito.it/."