A Sleep Study for ISP Networks: Evaluating Link Sleeping on Real World Data

Lukas Röllin*
ETH Zürich
roellinl@ethz.ch

Romain Jacob*
ETH Zürich
jacobr@ethz.ch

Laurent Vanbever*
ETH Zürich
lvanbever@ethz.ch

ABSTRACT

Turning off under-utilized network links is a promising energy-saving technique. In this paper, we present Hypnos, a link sleeping system targeted at low-utilization wired networks and assess its efficiency using real-world data from two European ISPs. In those two case studies, we find that Hypnos can turn off more than a third of all links without congesting the network. This confirms the promise of link sleeping in low-utilization networks.

CCS CONCEPTS

• **Networks** → **Physical links**; • **Hardware** → *Power estimation and optimization*; • **General and reference** → Empirical studies.

KEYWORDS

Sustainable Networking, Link Sleeping

ACM Reference Format:
Lukas Röllin, Romain Jacob, and Laurent Vanbever. 2024. A Sleep Study for ISP Networks: Evaluating Link Sleeping on Real World Data. In *Proceedings of 3rd Workshop on Sustainable Computer Systems (HotCarbon'24).* ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

In recent years, transceivers have increased in both capacity and power demand. Energy efficiency improves when transceivers are at 100% utilization [16], but, in practice, network links are often underutilized, even in datacenter networks [11]. Unlike servers, today's wired networks do not reduce their power draw by a lot when the utilization is low [15]. As a result, [11] suggests the network could make up around 20% of the IT power in a datacenter. This fraction is likely much larger in low-loaded networks such as ISPs. Hence, one could achieve important energy savings by improving energy proportionality in wired networks.

Energy proportionality is well-studied topic in the networking literature. In short, there are two classes of approaches: sleeping, *i.e.*, turning things off whenever possible, or rate adaptation, *i.e.*, setting the links to lower bitrates [20]. Rate adaptation is potentially more practical as it does not affect the routing topology; however, the power savings are limited by the fixed power cost to keep the link up, which sometimes dominates the total port power [15]. Besides, not all transceivers and network devices support multiple bitrates, which limits the generality of the approach.

In this paper, we focus on link sleeping, *i.e.*, turning links off. In a prior poster [24], we observed that transceivers can take multiple seconds to turn on and off. This renders sleeping at the traffic scale (as suggested in [20]) unfeasible. However, we argued one could still put links to sleep at longer timescales (*e.g.*, a couple of times per day) and proposed a first system prototype. This system performs four main functions:

(1) Collect network state information
(2) Select links to put to sleep
(3) Turn links off
(4) Wake links up on demand

Intuitively, the potential savings from sleeping strongly depend on the network load and degree of connectivity. Moreover, one may be more or less aggressive in the selection of links to put to sleep; turning more links off improves energy savings but is more likely to create congestion events on the remaining links. The likelihood of creating congestion depends on how stable the traffic demand is; the more bursty it is, the likelier it is the system makes "mistakes," *i.e.*, it turned off links that would have avoided congestion.

[24] described a first prototype but lacked a fundamental part in its evaluation: without access to real-world traffic and topology information, it could not assess how efficient the system would be at putting links to sleep; *i.e.*, how many links would it turn off, and how often would sleeping decisions lead to congestion?

This paper fills this gap. We present a refined link sleeping system called Hypnos and evaluate its efficiency on real-world data from two ISPs. We show that, despite its simple logic, Hypnos is very efficient; more specifically,

- Hypnos turns off more than a third of links in two real-world case study of lowly-loaded networks; (§ 3.4)
- it does so without causing congestion; (§ 3.4)
- it adapts well to high-load scenarios; (§ 3.5, § 3.8)
- it can be configured to maintain link failure resilience. (§ 3.6)

2 HYPNOS

2.1 Overall Design

Hypnos aims to put as many links to sleep as possible without disrupting the network. In other words, it has three objectives.

- It must *not disconnect the network* by putting links to sleep.
- It must *decide which links to put to sleep* while minimizing traffic redirection and congestion.
- If congestion happens, it must *react quickly* and turn links back on to resolve the congestion.

*The CRediT statement for this work is available in Appendix A.

HotCarbon'24, July 9, 2024, Santa Cruz, CA

While we aim to set as many links to sleep and keep them asleep for as long as possible, avoiding congestion takes priority. Hypnos' four main functions are described below and illustrated in Fig. 1.

(1) Collecting network state. One key design choice is deciding whether one link can be safely turned off. To know exactly where the current traffic would be re-routed, one would need the flow-level traffic matrix and complete routing state, which is very challenging to collect and process in real-time. Instead, Hypnos relies on a heuristic based on link loads (Fig. 1i), which are already commonly collected today at five-minute granularity via SNMP, similar monitoring protocols. Link loads can also be collected *e.g.*, via the OSPF TE Metric Extensions (RFC 7471) or estimated using tools like sFlow [22] or NetFlow [5].

(2) Selecting links to put to sleep. Link loads are aggregated to a centralized controller, which selects the links to turn off (Fig. 1ii). Hypnos uses a centralized controller to avoid accidental network partitions; the controller always knows which links are asleep and which must stay up to keep the network connected. In a decentralized system, guaranteeing connectivity becomes non-trivial.

Hypnos sets the total volume of traffic allowed to be rerouted with a *reroute budget* that lets the operator tune the acceptable amount of traffic moved to a different path due to sleeping.

(3) Turning links off. To limit traffic disruption, Hypnos turns links off in two stages: First, it increases the IGP link weight to drain all traffic from the link; then, if no congestion is observed on the other links, Hypnos turns the link off (Fig. 1iii).

(4) Waking up the network. Since avoiding congestion is prioritized over saving energy, Hypnos uses a conservative wake-up logic: if one of the network links becomes congested (Fig. 1iv), Hypnos immediately triggers the wake-up of all sleeping links (Fig. 1v).

2.2 Sleeping Decision Algorithm

The link sleeping decision algorithm is the core of Hypnos. It proceeds in three steps:

- defining sleeping link candidates;
- sorting candidates by increasing load;
- iteratively turning candidates off while checking connectivity and rerouting constraints.

To define link candidates, Hypnos checks all the link loads of the routers connected to a given link. As explained in § 2.1, we cannot easily know where the link traffic would be rerouted, so Hypnos uses the following heuristic. Take one candidate link and assume all of its traffic would be rerouted via the highest-loaded link on the same router; if that creates congestion, disregard the link as a sleeping candidate. Then, iterate over all links.

The second step sorts all candidate links by the traffic volume they carry, in bits per second. Using traffic volume instead of utilization (in %) is important in networks with different link capacities: turning off a 100G link at 1% results in a lot more rerouting than a 1G link at 50% utilization. Hence, Hypnos indirectly prioritizes turning off low-capacity links.

Finally, Hypnos iterates over the link candidates. It first checks that turning off the current candidate does not disconnect the network once accounting for the already turned-off links. If not, it then checks if there is enough reroute budget left for that link. The algorithm stops when one of the conditions no longer holds.

In addition, there is one special case that the algorithm handles first: Two routers are sometimes connected via multiple links, known as a bundle. Bundles are special because if one link is turned off, the link traffic is guaranteed to be rerouted to the remaining links from the bundle. Thus, Hypnos optimizes the sleeping of bundles by aggregating the bundle traffic over as few links as necessary to carry that traffic and putting all other links to sleep. Then, it aggregates the remaining links from the bundle as one virtual link and runs the link sleeping decision algorithm described above.[1]

3 EVALUATION

We now thoroughly evaluate how efficient Hypnos is at putting links to sleep in realistic settings; *i.e.*, how many links would it turn off, and how often would sleeping decisions lead to congestion?

Our ISP dataset (§ 3.1 [14]) and complete evaluation code [23] are both publicly available.

3.1 Dataset

Before diving into the evaluation, we present a short overview of the topologies we use to evaluate Hypnos' algorithm (Table 1).

Table 1: Overview of the evaluation's topologies

Dataset	Nodes	Links	Avg. link load	Avg. node degree
Switch	145	238	2.1%	3.28
SURF	462	745	1.2%	3.22
Repetita [9] (avg)	38.9	54.2	32.5%	2.64

We obtained the topology, IGP weights, and link load information from the production network of two ISPs: Switch (CH) and SURF (NL), which are both National Research and Education Networks. For our evaluation, we use the link load information of their internal links at a granularity of five-minute intervals.

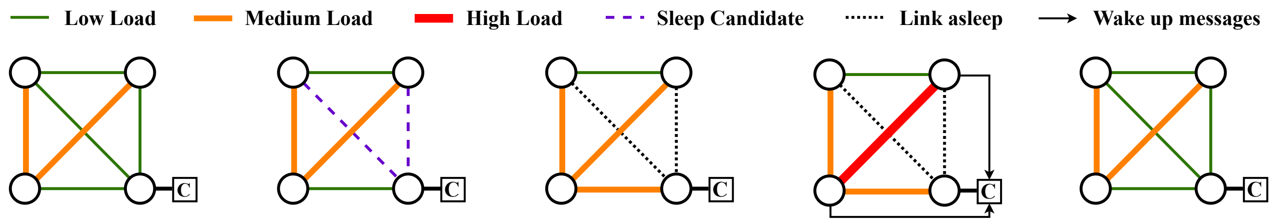
To test against a more diverse set of topologies, we also use Repetita [9]: a collection of real topologies and synthetic traffic matrices that is meant to compare traffic engineering (TE) algorithms. Since it is used for TE algorithms, the average link load is much higher than for the ISP networks, making it almost impossible to turn any link off. The evaluation on Repetita aims to show whether Hypnos creates problems in high-load scenarios.

Finally, we also use scaled versions of the traffic scenarios above. We scale the traffic matrix down for Repetita and scale up for the ISPs. For up-scaling, we set a cap on individual link loads to 70%.

3.2 Extracting the Traffic Matrix

Hypnos uses only network link loads as inputs (see § 2). However, to evaluate whether Hypnos creates congestion, we need the traffic matrix to compute accurately where traffic gets rerouted. To obtain this data, we use a technique called tomogravity [29] that approximates traffic matrices from the link loads.

[1] Conceptually, that is similar to performing rate adaptation at the bundle level, where the rate is adapted by selecting a subset of links between two nodes, rather than changing the physical layer.



(i) Collect link loads (ii) Select links to turn off (iii) Turn off links (iv) Wake up (v) Network awake

Figure 1: Visual representation of steps Hypnos performs to turn off and wake up the network

As the tomogravity method scales quadratically with the number of links, it became a bottleneck for our evaluation of SURF, which is much larger than Switch (Table 1). Thus, our evaluation uses 75 days of data for Switch but only 14 days for SURF. Note that the bottleneck comes from the evaluation needs, not from Hypnos. Running on a single core, Hypnos derives the list of links to turn off in less than 300ms for Switch and less than 3s for SURF.

3.3 Metrics

This paper focuses on Hypnos' algorithm for selecting links to put to sleep. Thus, the main performance metric we consider is the number of links that Hypnos turns off. This metric is upper bounded by the size of a network's spanning tree, since we want to keep the network connected. Therefore, the maximum number of links that Hypnos can turn off is 94 for Switch and 284 for SURF.

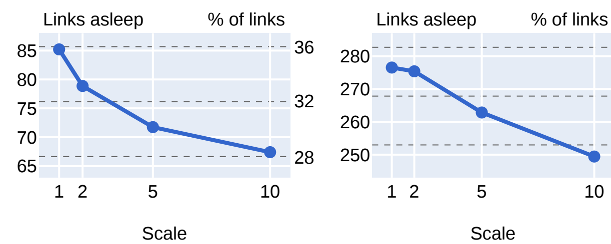
Conversely, Hypnos can create congestion in a network in two ways. The first case is when the link sleeping heuristic makes a bad decision, and too much traffic gets rerouted on a link that becomes congested. This can happen because the heuristic takes a local view on link loads (§ 2.1) while turning a link off can have global effects on routing. The second case is when turning a link off causes congestion on the *future* traffic demand; *i.e.*, when the demand changes significantly. This is a generic problem for any scheme that does not know nor predict future demands.

We measure Hypnos' negative impact as the number of 5-minute intervals where it leads to mild congestion (over 80%) or severe (over 100%) on at least one of the network links. Once mild congestion is detected, Hypnos wakes up the network immediately to avoid severe congestion. The threshold for "mild" congestion must be set high enough to avoid unnecessarily waking up the network but low enough to leave time to turn the links back on before severe congestion. We set the threshold for "mild" congestion somewhat arbitrarily as we have no information about fine-grained traffic load in the network nor on the time required to wake up the network.

3.4 Performance on ISP Networks

How many links does Hypnos turn off? Fig. 2 shows the mean number of links that can be turned off over the entire dataset. When Hypnos creates congestion in one 5-minute interval, we count the number of links put to sleep as zero for that interval since the entire network is being woken up.

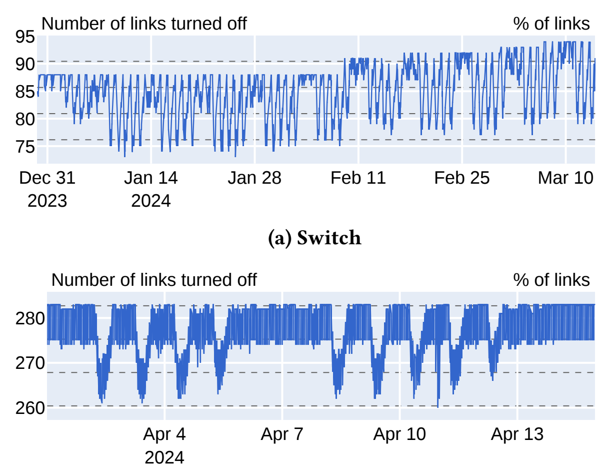
For Switch, Hypnos turns off 85 links on average, which is 36% of the network's links (Fig. 2a). As expected, when scaling the traffic up, the savings go down; but even at ten times the traffic, Hypnos still turns off around 67 links (28%). This is because, in our case studies, many links carry almost no traffic at all; thus, those links can still be turned off after scaling the traffic by 10x. The results are similar for SURF (Fig. 2b).



(a) Switch (b) SURF

Figure 2: Hypnos turns off more than 30% of the links, even when we artificially scale up the traffic load.

Does Hypnos adapt to changing load conditions? Fig. 3 shows that Hypnos' algorithm adapts nicely to the daily and weekly load patterns seen in the network and adjusts the number of links that can be turned off accordingly. Notably, we can see that for Switch, the number of links we can turn off increases, this is due to new links becoming available during the interval.



(b) SURF

Figure 3: Hypnos adapts to daily and weekly load patterns.

How stable is the list of sleeping links? Even if Hypnos limits the volume of rerouted traffic, one does not want links flapping on and off all the time. Fig. 4 shows the percentage of links that turn on or off in every timestep for SURF. We find that most of the 278 sleeping links stay asleep, and only around 12% of those have to change their state. We observe similar results for Switch (not shown), where around 10% of links change their state between two intervals. This shows that most links remain stable; the routing protocol needs to handle only a few link state changes in every five-minute interval.

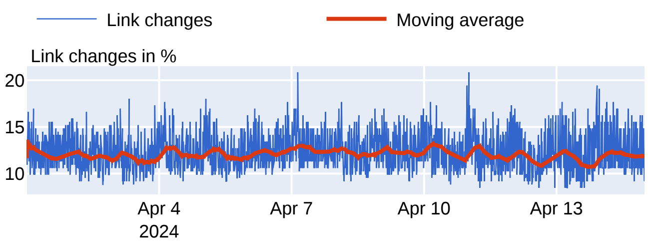
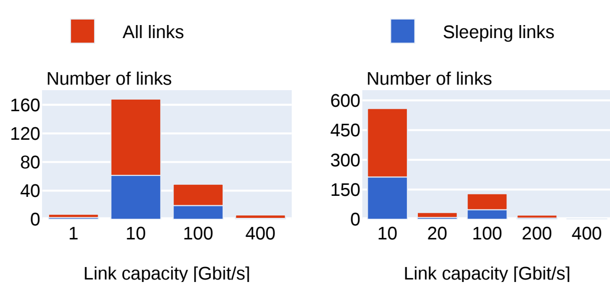


Figure 4: Only a fraction of links change their state between intervals, reducing the amount of rerouting that is necessary.

Which links are put to sleep? Since we have links with different capacities, one might expect Hypnos' algorithm to turn off only the low-capacity ones. We found that this is not the case; as shown in Fig. 5, Hypnos also turns off high-capacity links when possible. For Switch, it turns off one out of the six 400 Gbps links on average without creating congestion. The results are similar for SURF.[2]



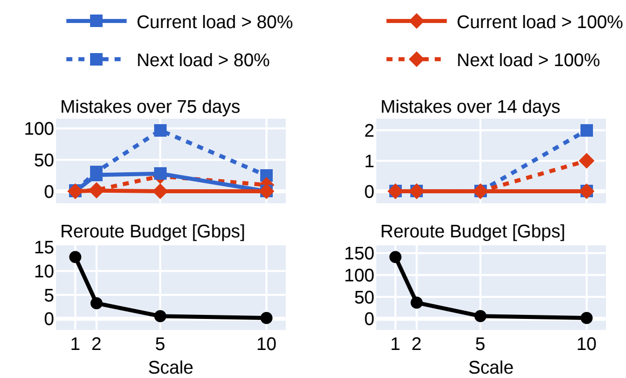
(a) Switch **(b) SURF**

Figure 5: Hypnos does not just turn off low-capacity links.

How many mistakes does Hypnos make? Fig. 6 shows the number of mistakes made by Hypnos as the load increases. We count one mistake for each timestep where mild (blue lines) or severe (red lines) congestion happens on one of the links. The plots also distinguish congestion events in the current (solid lines) and in the next (dashed lines) timesteps. Congestion in the next timestep may occur when traffic conditions change significantly in consecutive timesteps. The bottom part of the plot shows the average reroute budget given by Eq. (1).

With unscaled traffic, Hypnos never creates congestion in the current timestep. A few times, traffic demand changes sufficiently during the next five-minute interval to result in congestion. When the traffic changes more gradually, Hypnos avoid congestion by waking up the network as the load builds up.

As expected, we see that increasing the network load makes it harder for Hypnos to turn off links without making mistakes. However, once we scale up the load even more, the number of mistakes decreases again, as Hypnos can put fewer links to sleep.



(a) Switch **(b) SURF**

Figure 6: Hypnos has to be more cautious for higher loads.

This behavior is partly due to the dynamic nature of the reroute budget (§ 3.5). When the load in the network increases, we decrease the reroute budget as it is easier to make mistakes at higher loads. For Switch, it looks like the decrease does not happen fast enough, so we have more mistakes at scale 5 than at scale 10. This result suggests that it might be possible to tune the reroute budget to avoid creating congestion regardless of the load.

3.5 Reroute budget

The reroute budget is a key parameter of Hypnos. If we set the budget too low, there are no savings because Hypnos has no leeway to turn any links off. If we set it too high, Hypnos turns off too many links and is more likely to cause congestion in the network.

In this paper, we set the reroute budget as follows.

$$reroute\_budget = \frac{sum\_link\_capacity}{250} * \frac{0.0001}{network\_util^2} \quad (1)$$

The first term of the formula estimates the order of magnitude of traffic the network experiences. A reroute budget of 10 Gbit might be acceptable for a network with terabits of traffic but is probably not ideal for a network only with a few gigabits of traffic.

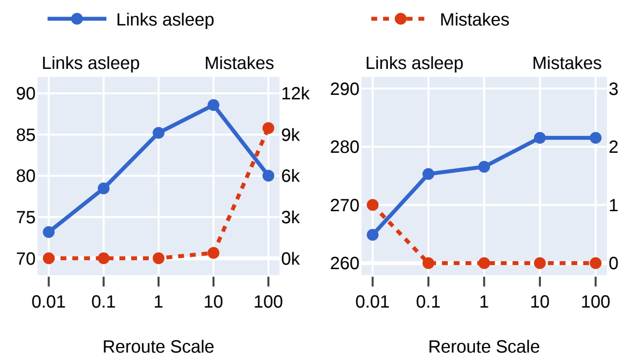
The second term captures the network load; we want the reroute budget to be small if the load is high and vice versa. If the network's average network utilization is 1%, the second factor becomes 1.

The reroute budget defined in Eq. (1) is not meant to be optimal; we found it to work well for our case studies, but it might not work in other scenarios.

Fig. 7 shows how Hypnos behaves if we scale up or down the reroute budget. For Switch, we can clearly see that increasing the budget causes Hypnos to make many more mistakes while reducing it limits the savings. Since we wake up the network anytime we make a mistake, the savings also decrease with higher reroute budgets. Interestingly, for SURF, we only see similar behavior when

[2]One may note some unusual link speeds in Fig. 5b. This is because we do not have data on each physical link for SURF; some of the link data are bundled, resulting in capacities of, *e.g.*, 20 Gbps, which are just two 10 Gbps interfaces bundled together.

reducing the reroute budget; when increasing the reroute budget, nothing really changes. We believe this is due to the lower load of SURF, which means the algorithm is not limited by the reroute budget but by the connectivity constraint.



(a) Switch **(b) SURF**

Figure 7: Switch is more sensitive to budget changes.

It is not clear how to optimally set the reroute budget for a given network and traffic load. Currently, the simplest approach is to simulate the sleep decision algorithm's behavior over common load scenarios and tune the reroute budget accordingly.

3.6 Two connectedness

Until now, we only considered that the network should remain connected. Since the traffic load is very low (Table 1), Hypnos almost reduces the active topology to a spanning tree. This might be considered an overly aggressive sleeping strategy, as any failure could momentarily disconnect the network.

Therefore, we also evaluate a version of Hypnos that aims to keep the network 2-connected at all times. However, since Switch and SURF networks are not 2-connected to begin with, we constraint Hypnos to keep the largest component of the network 2-connected. The results are summarized in Table 2.

Table 2: Hypnos can still turn some links off even with a 2-connectedness constraint.

Constraint	1-connected (default)	2-connected
Switch	85 (36%)	43 (18%)
SURF	280 (38%)	52 (7%)

For Switch, enforcing 2-connectedness reduces the number of sleeping links from 85 down to 43 and, for SURF, from 280 to 52 (Table 2). Thus, even with a 2-connectedness constraint, Hypnos can still turn off a few percent of the network links.

If the time to react to failures is relaxed we can even provide both high savings and "delayed" 2-connectedness. Since all of the links are still there but just turned off, one can instruct the system to wake up the network in case of a failure. Then the network can reconnect itself within the time it takes to wake up the interfaces. Of course, it is only possible to reconnect the network if the awake topology was 2-connected to begin with. In previous work [24], we have shown that the interface wake-up time, depending on its type, is in the order of a few seconds to a few 10s of seconds.

3.7 Power savings

Ultimately, Hypnos aims to save energy. Modeling the power drawn by routers is not trivial [1, 13, 19, 26, 27]. The power savings from link sleeping are hard to estimate because turning off a port affects many subsystems (transceivers, DSPs, SerDes, etc.). To date, there are no fine-grained power models available for modern routers.

However, we can approximate the potential gains on *transceiver power* rather easily. Table 3 lists datasheet power numbers for different speeds of optical long-range (LR) transceivers.[3]

Table 3: Approximate power numbers for LR transceivers

Capacity	1G	10G	100G	400G
Power	1W	1W	4W	10.5W

Multiplying the power numbers from Table 3 with the number and types of links that Hypnos turns off (Fig. 5), we find that Switch and SURF could save on average 307W and 943W on transceiver power, respectively. This translates into energy savings of 2.7MWh and 8.3MWh per year, respectively.

3.8 Performance on Repetita

To show that Hypnos is not overfitted to our ISP case studies, we evaluate its performance on Repetita [9], which provides around 250 different test scenarios. It consists of real network topologies with synthetic traffic matrices intended to benchmark traffic engineering (TE) algorithms. For our evaluation, we let Hypnos' algorithm run on all topologies and traffic matrices to determine how many links can be turned off and, more importantly, how often sleeping is causing congestion.

Table 4: Hypnos saves little but does no harm under high load.

Scale	Avg. link load	Links turned off	Link loads over 80%	Link loads over 100%
10%	3.2%	7.25	0	0
50%	16.2%	1.37	0	0
100%	32.5%	0.53	0	0

We find that for all the topologies, Hypnos does not make any mistakes (Table 4). Hypnos does not cause any additional congestion in the network, even though the dataset provided is specifically highly loaded to allow for comparing traffic engineering algorithms (Table 1: Repetita's average load is about 15 times higher than that of Switch). While there is not much saving possible at those loads,[4] Hypnos adapts well to high loads and does not cause problems.

[3]LR transceivers are capable of transmitting up to 10km. We use the power numbers of Flexoptics LR transceivers [7] because they are the most common type of transceiver used in Switch.

[4]Another reason for the smaller savings is that many of the networks in Repetita are not well connected, so there is not as much potential to turn links off. Even without any traffic, the maximum number of links one could turn off is 16.3 out of 54.2.

4 RELATED WORK

Putting network links to sleep to save energy is an old idea, popularized by Gupta and Singh in 2003 [10] and then analytically studied by Nedevschi et al. [20]. A few years later, ElasticTree [12] applied link sleeping to data center networks; the key insight there was that one can leverage the symmetrical topology of fat trees to simplify the optimization problem of deciding which links can be safely turned off. While promising, this line of work suffers from a fundamental practical limitation: link sleeping at traffic timescales is made unfeasible by the time required to turn links back up, which is still measured in seconds today [24].

The other main approach for energy savings is rate adaptation. One method, also studied in [20], is to change the physical layer standard used (*e.g.*, setting a 10Gbps port to run at 1Gbps). This technique, practically explored in [15], is interesting as it does not change the network topology as link sleeping does. However, the potential savings are more limited in today's network as only a few link speeds exist, and rate adaptation is only possible if both link endpoints support the same speeds. Another rate adaptation method is to opportunistically throttle the link, as popularised by the Energy Efficient Ethernet standard (EEE or IEEE 802.3az, [4]). EEE essentially works by disabling the SerDes for short intervals without traffic; this approach no longer works for optical communication, as the light has to stay "on" all the time, leading to EEE's obsolescence. In the optical domain, RADWAN [25] explores new ways of implementing rate adaptation, though the work aims to maximize bandwidth, not energy savings. The way Hypnos handles bundled links (§ 2.2) can be seen as a form of rate adaptation.

Many works suggested integrating sleeping into the IGP protocols [2, 6] or as part of the objective function of a traffic engineering problem [3, 3, 17, 18, 21, 28]. Those prior works are very similar to Hypnos in spirit, and all share one limitation: realistically estimating the potential energy savings is impossible without accurate power models. We know what can be turned off for a decade, but we still do not know whether the energy savings are worth it—but we are working on it! Hypnos differentiates itself by its focus on practicality; *e.g.*, its design assumes realistic wake-up times and does not require modifying existing routing protocols.

Finally, Hypnos avoids packet drops when turning links off by draining the links first. This paper does not detail this point as it is a well-understood problem [8]. In essence, one can guarantee that no micro-loops occur by incrementally changing IGP link weights, which takes time. This is not a major concern for Hypnos, as it is designed to turn links off at large timescales (minutes or hours).

5 CONCLUSION AND FUTURE WORK

With this paper, we confirm that, in lowly loaded networks such as ISPs, it is possible to turn off a large number of links (§ 3.4 : >30%). We show that a simple heuristic (§ 2.2) to select the links to put to sleep appears good enough to avoid creating congestion on realistic network topologies and traffic demands. We materialize these observations with a link sleeping system prototype called Hypnos. The potential energy savings from link sleeping appear modest (§ 3.7), but we argue that any savings should be considered if they come with minimal downsides.

Hypnos appears already very effective at putting networks to sleep. However, several design points must be optimized further to make link sleeping really *too-easy-not-to-do-it*.

Optimizing the reroute budget The reroute budget is a key parameter of Hypnos that must be tuned to a given network capacity and traffic load (§ 3.5). How to optimally set it is an important question that deserves further attention.

Predicting traffic demands Most of the congestion events triggered by Hypnos result from assuming that traffic demands remain constant over the next five-minute interval. It would be natural to enhance the system with some mechanism to predict future demands and/or learn from past mistakes.

Optimizing link state stability Currently, Hypnos starts every link decision-making process anew; *i.e.*, it does not consider which links are already asleep. Because the algorithm is deterministic and the link loads are fairly stable in our case studies, this leads to a few links changing state at each iteration (Fig. 4). It should be straightforward to extend Hypnos to minimize the number of state changes, which would help make the system more practical.

Quantifying the latency cost In this paper, we only consider the "cost" of link sleeping in terms of congestion events. However, there can also be a latency cost for traffic that gets rerouted along a longer path. Estimating the impact of sleeping on latency is challenging and its practical relevance for ISP traffic is unclear. Nonetheless, latency should be considered.

Optimizing the transient While this is beyond the scope of this paper, an important aspect of a link sleeping system is managing the transient network state; *i.e.*, what happens while links are being turned on or off. To turn off links we just need to wait until the IGP has drained the link of traffic. When turning a link back on, we try to resolve an issue as quickly as possible, which is time-sensitive. In [24], we showed that transceivers can bottleneck the wake-up time of the network. To improve the transient, one needs to look at improving the turn-on time of transceivers.

Optimizing for power To date, we lack accurate models predicting the effect of link sleeping on routers' power demand; deriving fine-grained network power models is part of our ongoing research efforts. With such models, Hypnos could be extended to turn off links that optimize power savings.

In short, we showed that link sleeping is feasible in real-world scenarios and worth future work to optimize its performance further.

ACKNOWLEDGMENTS

We would like to thank Simon Leinen from Switch and Marijke Kaat and Edwin Verheul from SURF for their help in gathering the topology and traffic data used in this study.

This work was supported by the ETH Future Computing Laboratory (EFCL), financed by a donation from Huawei Technologies.

REFERENCES

[1] Jaewon Ahn and Hong-Shik Park. 2014. Measurement and Modeling the Power Consumption of Router Interface. In *16th International Conference on Advanced Communication Technology*. 860–863. https://doi.org/10.1109/ICACT.2014.6779082

[2] Aruna Prem Bianzino, Luca Chiaraviglio, Marco Mellia, and Jean-Louis Rougier. 2012. GRiDA: GReen Distributed Algorithm for Energy-Efficient IP Backbone Networks. *Computer Networks* 56, 14 (Sept. 2012), 3219–3232. https://doi.org/10.1016/j.comnet.2012.06.011

[3] Luca Chiaraviglio, Marco Mellia, and Fabio Neri. 2012. Minimizing ISP Network Energy Cost: Formulation and Solutions. *IEEE/ACM Transactions on Networking* 20, 2 (April 2012), 463–476. https://doi.org/10.1109/TNET.2011.2161487

[4] Ken Christensen, Pedro Reviriego, Bruce Nordman, Michael Bennett, Mehrgan Mostowfi, and Juan Antonio Maestro. 2010. IEEE 802.3az: The Road to Energy Efficient Ethernet. *IEEE Communications Magazine* 48, 11 (Nov. 2010), 50–56. https://doi.org/10.1109/MCOM.2010.5621967

[5] Benoît Claise. 2004. *Cisco Systems NetFlow Services Export Version 9*. Request for Comments RFC 3954. Internet Engineering Task Force. https://doi.org/10.17487/RFC3954

[6] M. D'Arienzo and S. P. Romano. 2016. GOSPF: An Energy Efficient Implementation of the OSPF Routing Protocol. *Journal of Network and Computer Applications* 75 (Nov. 2016), 110–127. https://doi.org/10.1016/j.jnca.2016.07.011

[7] FLEXOPTIX. 2024. FLEXOPTIX. https://www.flexoptix.net/

[8] P. Francois, M. Shand, and O. Bonaventure. 2007. Disruption Free Topology Reconfiguration in OSPF Networks. In *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*. 89–97. https://doi.org/10.1109/INFCOM.2007.19

[9] Steven Gay, Pierre Schaus, and Stefano Vissicchio. 2017. REPETITA: Repeatable Experiments for Performance Evaluation of Traffic-Engineering Algorithms. https://doi.org/10.48550/arXiv.1710.08665 arXiv:1710.08665 [cs]

[10] Maruti Gupta and Suresh Singh. 2003. Greening of the Internet. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '03)*. Association for Computing Machinery, New York, NY, USA, 19–26. https://doi.org/10.1145/863955.863959

[11] Haiyang Han, Nikos Terzenidis, Dimitris Syrivelis, Arash F. Beldachi, George T. Kanellos, Yigit Demir, Jie Gu, Srikanth Kandula, Nikos Pleros, Fabián Bustamante, and Nikos Hardavellas. 2021. Energy-Proportional Data Center Network Architecture Through OS, Switch and Laser Co-design. https://doi.org/10.48550/arXiv.2112.02083 arXiv:2112.02083 [cs]

[12] Brandon Heller, Srini Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. 2010. ElasticTree: Saving Energy in Data Center Networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*. USENIX Association, USA, 17. https://www.usenix.org/legacy/event/nsdi10/tech/full_papers/heller.pdf

[13] Helmut Hlavacs, Georges Da Costa, and Jean-Marc Pierson. 2009. Energy Consumption of Residential and Professional Switches. In *2009 International Conference on Computational Science and Engineering*, Vol. 1. 240–246. https://doi.org/10.1109/CSE.2009.244

[14] Romain Jacob. 2024. ISP Topology and Traffic Dataset. https://doi.org/10.5281/zenodo.12580396

[15] Romain Jacob, Jackie Lim, and Laurent Vanbever. 2023. Does Rate Adaptation at Daily Timescales Make Sense?. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*. ACM, Boston MA USA, 1–7. https://doi.org/10.1145/3604930.3605713

[16] Itzik Kiselevsky. 2023. Evolution of Switches Power Consumption. https://eng.ox.ac.uk/media/11vdkdtb/itzikk_evolution-of-switches-power-consumption.pdf

[17] Youcef Magnouche, Jérémie Leguay, and Feng Zeng. 2023. Safe Routing in Energy-aware IP Networks. In *2023 19th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, Vilanova i la Geltru, Spain, 1–8. https://doi.org/10.1109/DRCN57075.2023.10108152

[18] Priya Mahadevan, Sujata Banerjee, and Puneet Sharma. 2010. Energy Proportionality of an Enterprise Network. In *Proceedings of the First ACM SIGCOMM Workshop on Green Networking (Green Networking '10)*. Association for Computing Machinery, New York, NY, USA, 53–60. https://doi.org/10.1145/1851290.1851302

[19] Hakim Mellah and Brunilde Sansò. 2011. Routers vs Switches, How Much More Power Do They Really Consume? A Datasheet Analysis. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. 1–6. https://doi.org/10.1109/WoWMoM.2011.5986484

[20] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. 2008. Reducing Network Energy Consumption via Sleeping and Rate-Adaptation. In *5th USENIX Symposium on Networked Systems Design and Implementation (NSDI 08)*. https://www.usenix.org/conference/nsdi-08/reducing-network-energy-consumption-sleeping-and-rate-adaptation

[21] Tian Pan, Xiaoyu Peng, Qianqian Shi, Zizheng Bian, Xingchen Lin, Enge Song, Fuliang Li, Yang Xu, and Tao Huang. 2021. GreenTE.Ai: Power-Aware Traffic Engineering via Deep Reinforcement Learning. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. 1–6. https://doi.org/10.1109/IWQOS52092.2021.9521281

[22] Sonia Panchen, Neil McKee, and Peter Phaal. 2001. *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks*. Request for Comments RFC 3176. Internet Engineering Task Force. https://doi.org/10.17487/RFC3176

[23] Lukas Röllin. 2024. Nsg-Ethz/Hypnos: Release 1.0. Zenodo. https://doi.org/10.5281/zenodo.12582811

[24] Lukas Röllin, Romain Jacob, and Laurent Vanbever. 2023. Poster: What Keeps Your Network up at Night?. In *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT 2023)*. Association for Computing Machinery, New York, NY, USA, 59–60. https://doi.org/10.1145/3624354.3630092

[25] Rachee Singh, Manya Ghobadi, Klaus-Tycho Foerster, Mark Filer, and Phillipa Gill. 2018. RADWAN: Rate Adaptive Wide Area Network. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 547–560. https://doi.org/10.1145/3230543.3230570

[26] Arun Vishwanath, Kerry Hinton, Robert W. A. Ayre, and Rodney S. Tucker. 2014. Modeling Energy Consumption in High-Capacity Routers and Switches. *IEEE Journal on Selected Areas in Communications* 32, 8 (Aug. 2014), 1524–1532. https://doi.org/10.1109/JSAC.2014.2335312

[27] Lane Wigley. 2022. Assessing the Real-World Environmental Impact of Routers. https://blogs.cisco.com/sp/assessing-the-real-world-environmental-impact-of-routers

[28] Mingui Zhang, Cheng Yi, Bin Liu, and Beichuan Zhang. 2010. GreenTE: Power-aware Traffic Engineering. In *The 18th IEEE International Conference on Network Protocols*. 21–30. https://doi.org/10.1109/ICNP.2010.5762751

[29] Yin Zhang, Matthew Roughan, Nick Duffield, and Albert Greenberg. 2003. Fast Accurate Computation of Large-Scale IP Traffic Matrices from Link Loads. *ACM SIGMETRICS Performance Evaluation Review* 31, 1 (June 2003), 206–217. https://doi.org/10.1145/885651.781053

A CREDIT STATEMENT

This section lists the author's contributions to this work. The contributions are described using CRediT, the Contributor Roles Taxonomy, an ANSI/NISO standard.

All authors agree with this declaration of contributions.

Lukas Röllin	0009-0005-0720-5134
Data curation	Supporting
Formal analysis	Lead
Investigation	Lead
Methodology	Lead
Software	Lead
Validation	Equal
Visualization	Lead
Writing – original draft	Equal
Writing – review & editing	Lead

Romain Jacob	0000-0002-2218-5750
Conceptualization	Lead
Data curation	Lead
Formal analysis	Supporting
Funding acquisition	Equal
Investigation	Supporting
Methodology	Supporting
Project administration	Lead
Software	Supporting
Supervision	Lead
Validation	Equal
Visualization	Supporting
Writing – original draft	Equal
Writing – review & editing	Supporting

Laurent Vanbever	0000-0003-1455-4381
Conceptualization	Supporting
Funding acquisition	Equal
Methodology	Supporting
Resources	Lead
Supervision	Supporting
Writing – original draft	Supporting