Routing and Scheduling for Energy and Delay Minimization in the Powerdown Model

Matthew Andrews
Bell Labs
Murray Hill, NJ
andrews@research.bell-labs.com

Antonio Fernández Anta
U. Rey Juan Carlos
Móstoles, Madrid, Spain
anto@gsyc.es

Lisa Zhang
Bell Labs
Murray Hill, NJ
ylz@research.bell-labs.com

Wenbo Zhao
UCSD
La Jolla, CA
w3zhao@ucsd.edu

***Abstract*—Energy conservation is drawing increasing attention in data networking. One school of thought believes that a dominant amount of energy saving comes from turning off network elements. The difficulty is that transitioning between the active and sleeping modes consumes considerable energy and time. This results in an obvious trade-off between saving energy and provisioning performance guarantees such as end-to-end delays.**

We study the following routing and scheduling problem in a network in which each network element either operates in the full-rate active mode or the zero-rate sleeping mode. For a given network and traffic matrix, routing determines the path along which each traffic stream traverses. For frame-based periodic scheduling, a schedule determines the active period per element within each frame and prioritizes packets within each active period. For a line topology, we present a schedule with close-to-minimum delay for a minimum active period per element. For an arbitrary topology, we partition the network into a collection of lines and utilize the near-optimal schedule along each line. Additional delay is incurred only when a path switches from one line to another. By minimizing the number of switchings via routing, we show a logarithmic approximation for both energy consumption and end-to-end delays.

If routing is given as input, we present two schedules one of which has active period proportional to the traffic load per network element, and the other proportional to the maximum load over all elements. The end-to-end delay of the latter is much improved compared to the delay for the former. This demonstrates the trade-off between energy and delay.

I. Introduction

The telecommunication infrastructure in the US is estimated to consume 60 billion Kwh per year. Such a enormous consumption partially results from the fact that most networks are engineered to handle peak traffic. Network elements tend to operate at full speed and consume maximum power, while typical traffic is only a small fraction of the maximum throughput. It is estimated that, if the energy consumption of each network element is adapted to be proportional to its traffic load, up to 80% of the energy in the access layer and up to 40% in the network core can be saved, totaling around 24 billion Kwh per year [1].

Powering down is a promising mechanisms for dynamically adapting the power consumption to the actual load at each network element (routers, switches, CPUs, Ethernet links, etc). It saves energy by switching off the element when possible. In this model each network element either operates in the active mode at the full rate or in the sleep mode at the zero rate. We consider provisioning a set of connections each with a desired connection rate in a network, with two potentially conflicting objectives of minimizing the total energy consumption by the network elements and the end-to-end delay experienced by the connections. Routing and scheduling are two integral components of the problem. Routing determines which path each connection follows, and scheduling decides the active periods for each network element and prioritizes packets within each active period. If switching between the two modes were free, then one plausible approach for scheduling would be activating a network element instantaneously at the arrival of each packet, processing at full rate till the queue drains and then switching to the sleep mode instantaneously. This effort tends to favor both energy and delay.

Unfortunately, switching between the active and sleep modes consumes considerable energy and time (it could go up to the order of milliseconds [6], [9]). Therefore, from the perspective of energy, the fraction of the total active time of a network element should be proportional to the total connection rates that the element carries. In addition, if the active proportion is the same, it consumes less energy to have long active periods infrequently than to have short active periods frequently, as this saves transition costs. On the other hand, from the perspective of delay, it is most convenient to operate in the active mode all the time. Barring this option, for the same active proportion it is advantageous to have short active periods frequently than long active periods infrequently, since the long sleep periods that accompany the long active periods contribute to the queuing delay.

To our knowledge, integral approaches to globally optimize power consumption at a network level are scarce. One of the papers that studies techniques of energy saving at a network level is due to Nedevschi et al. [8]. In this paper, changing the transmission rate and powering down (parts of) the network equipment are explored as two alternative techniques to globally reduce energy consumption in a network. When using sleep states, much time and energy can be used to transit from sleeping to active and vice versa if many such transitions occur. Then, the authors propose that edge routers group packets of the same source-destination pair and transmit them in bursts, in order to reduce the number of transitions and maximize the sleeping time.

II. Model and Results

We are given a network to support a set of connections, each of which needs to transmit packets between a pair of terminal

nodes at a specified rate. Connection i is routed along path P_i, where, P_i can be either given as input or be computed as part of the output. We study both situations. If the routes are not part of the input, the routing part of the problem specifies a route P_i for each connection i. The packet arrival is leaky-bucket controled and connection i has a desired rate of ρ_i. The scheduling part of the problem specifies the active period of each link, i.e. when a link operates at the full rate normalized to $R = 1$, and how packets are prioritized within each active period. Transitioning between two modes takes δ units of time, and consumes energy at the same rate as the full rate, though it cannot process any packets during the transition.

We focus on frame-based schedules, in which each link is active for a fixed duration within every frame of T steps, $(0, T]$, $(T, 2T], \cdots$. Let $R_e = \sum_{i:e\in P_i} \rho_i$ be the total rates that go through link e. Let

$$A_e = T \cdot R_e$$

be the minimum length of an active period for link e. We call a schedule *minimal* if the duration of the active period on each link e is exactly A_e. Obviously, for a fixed set of routes and for a fixed T, a minimal schedule is energy optimal.

We prove the following results. For simplicity, we assume that transmitting one packet across a link takes one unit of time, or *time step*. Our results below generalize to links with arbitrary link transmission times.

- We begin with a line topology. Since routing is fixed, we focus on scheduling only. We present a minimal schedule in which the end-to-end delay for each connection i of k_i links is upper bounded by $2T + k_i$. We also present a *quasi-minimal* schedule in which the duration of the active period is $A_{\max} = \max_e A_e$ for all e. For this case we show an end-to-end delay of $T + k_i$ for all i.
  For each fixed T, the energy-delay trade-off between these two schedules is obvious, since the one with smaller delay consumes more energy and vice versa. Within each schedule, we also observe the trade-off since a larger T implies a larger delay bound but fewer active-sleeping transitions, and therefore less energy consumption.
- We then show how to apply our techniques for line scheduling to obtain results for networks with arbitrary topology. If routing is not part of the input we first choose routes with the objective of minimizing $\sum_e A_e$ together with the transition cost. This sum can be approximated to within a factor of $\tilde{O}(\log n)$ where n is the size of the network. In addition, we can also guarantee that the routes form a tree. Via caterpillar decomposition we partition the tree into a collection of lines and each connection path intersects with at most $2 \log n$ lines. Along each line we adopt the minimal schedule. When a packet moves from line to line, a maximum extra delay of T is sufficient. This allows us to prove an $\tilde{O}(\log n)$ guarantee in the end-to-end delay for all connections.
- If routing is already specified, we present a straightforward minimum schedule with a delay bound of $T \cdot k_i$ for each connection i. However, if links are active for longer in sufficiently large frames, then the delay can be much improved. This again demonstrates an obvious trade-off between energy and delay.

III. Scheduling over a Line

We begin with scheduling a set of connections over a line. Note that the terminals of each connection are not necessarily the end points of the line. Although the line topology is restricted, the solution is used as a building block for handling arbitrary topologies. For a line, we can choose any frame size T, as long as $T \geq A_e + 2\delta$ for all e. Since $A_e = T \cdot R_e$, this implies

$$T \geq \max_e \frac{2\delta}{1 - R_e}.$$

Lemma 1. *For a line topology, after an initial delay of at most T each packet can reach its destination with at most $A_{\max}$ additional queueing delay. The activation period is A_e, minimal for each link e.*

Proof: We first define a minimal schedule. Let us label the links along the line as e_1, e_2, etc. For any integer $j \geq 1$, the jth active period for the ith link e_i is $[jT+i, jT+i+A_{e_i})$. From now on we consider a fixed j. Let S_i be the set of packets that have e_i as their first link and that are injected during the period $[(j-1)T + i, jT + i)$. We show in the following that the packets in $\cup_i S_i$ travel to their destinations during the jth active period of each link.

We assign to these packets timeslots in the active periods such that i) for each packet the timeslots over its routing path are non-decreasing; ii) for two packets that share a common link, they are assigned distinct slots over this common link. With these requirements, the timeslots define when each packet gets to advance.

From packets in $\cup_i S_i$, we repeatedly extract those whose paths form a minimal cover of the entire line, i.e. any proper subset of these packet paths cannot cover the entire line. From this minimal cover C, we removed overlaps and create an exact cover in which each link is covered by exactly one packet path. Since C is minimal, we start with the unique packet path, say p, that contains e_1. We walk along p until it first overlaps with another packet path, say q. We keep $p - p \cap q$ for the exact cover and remove $p \cap q$. Note that due to the minimality of C, q again is unique. We proceed to walk along q until it overlaps with another packet path. We remove the overlap from q in a similar manner. When all overlaps are removed, C becomes an exact cover. Now every path C is assigned the next available timeslot from each of the active periods. Note that the next available timeslot is the kth slot of each active period for some common k. This invariant holds through a simple induction. Note also that if a packet path is cut, the removed part always follows the unremoved part and is therefore always assigned a larger timeslot later on. This ensures the timeslot assignment is feasible for packet movement. When the union of the remaining packet paths do not cover the entire line, we carry out the above process on each disconnected line. Note also that at most A_e of packets in $\cup_i S_i$ require each link e.

The active periods therefore have enough slots for all packets. Hence, in addition to transmission time (reflected by the shifts of the active periods from one link to the next) packets in $\cup_i S_i$ experience a total of at most $A_{\max}$ queueing time.

The above centralized algorithm that iteratively creates covers can easily be replaced by a distributed protocol called *farthest-to-go* (FTG). For each link e, during its active period FTG gives priority to the packet in $\cup_i S_i$ that has the most number of remaining links to traverse. To see that FTG fits the proof above, we note that for each $k \geq 0$, the packets that use the kth timeslot in each active interval have their packets paths form a minimal cover of the line. ■

If we relax the active period of each link to be $A_{\max}$, each packet experiences no queueing delay once it starts moving, after an initial delay of up to T.

Lemma 2. *For a line topology, after an initial delay of at most T each packet can reach its destination with no further queueing delay. The activation period is $A_{\max}$ for each link.*

Proof: For any integer $j \geq 1$ the jth active period for the ith link e_i along the line is $[jT + i, jT + i + A_{\max})$. As in the proof of Lemma 1 let S_i be the set of packets that have e_i as their first link and that are injected during the period $[(j-1)T+i, jT+i)$. We show in the following that the packets in $\cup_i S_i$ travel to their destinations during the jth active period of each link.

Note that at most $A_{\max}$ of the packets from $\cup_i S_i$ have e along their paths. Therefore, the well-known interval graph coloring result (e.g. [5]) implies that each packet in S can be assigned a color in $[1, A_{\max}]$ such that two packets are assigned distinct colors if they share any link in common. These colors define the timeslots during the active periods that the packets can advance. In this way, when a packet moves from one link to the next it has a slot immediately for transmission. ■

IV. Combined Routing and Scheduling

We now turn our attention to the case of an arbitrary network topology. If routing is not given as input, we first choose a path P_i for each connection i. Recall that $R_e = \sum_{i:e \in P_i} \rho_i$ is the total rate on link e as a result of routing. Let

$$f(R_e) \quad = \quad \begin{cases} 0 & \text{for } R_e = 0 \\ 2\delta + R_e T & \text{for } R_e > 0 \end{cases} .$$

By routing connections with the objective of $\min \sum_e f(R_e)$ we minimize the total active periods together with the transition time over all links, and therefore minimize the energy. Note that the above formulation only makes sense when $R_e < 1$. We can reasonably assume $R_e \ll 1$ regardless of routing since this paper is motivated by the scenario in which full rate $R = 1$ is significantly larger than what is needed.

Note that $f(\cdot)$ is a concave function and the routing problem in fact corresponds to the well-studied buy-at-bulk network design problem. Awerbuch and Azar [3] offer a solution using the concept of *probabilistically approximating* metrics by tree metrics (see e.g. [4]. The idea is to approximate distances in a graph G by a probability distribution over trees. In particular, let $d_{ij}(G)$ be the length of link (i, j) in the graph G. Suppose that tree H is randomly chosen according to the probability distribution and let $d_{ij}(T)$ be the distance between i and j in the tree H. We say that the probability distribution over trees α-*probabilistically approximates* the graph G if,

$$E(d_{ij}(H))/\alpha \leq d_{ij}(G) \leq d_{ij}(H).$$

Awerbuch and Azar [3] show that an α stretch implies an $O(\alpha)$ approximation for buy-at-bulk if we randomly pick a tree according to the probability distribution and by then route all connections along the unique path specified by the tree. We use the following result on α.

Theorem 3 ([2]). *Every network G can be α-probabilistically-approximated by a polynomially computable probability distribution over* spanning *trees, for $\alpha = \tilde{O}(\log n)$.*

The notion of $\tilde{O}(\log n)$ hides terms $\log \log n$ and smaller. In fact the exact value of α is $O(\log n \log \log n \log^3 \log \log n)$. This implies,

Theorem 4. *There is a randomized algorithm that chooses a routing such that the total active period together with the transition time is $\tilde{O}(\log n)$ times the minimum optimal. In addition, the routes form a tree, and for any connection the expected routing distance over the tree is $\tilde{O}(\log n)$ times the shortest path distance over the original network.*

We further take advantage of the fact that the resulting routes form a tree. We solve the scheduling problem by combining caterpillar decomposition of a tree and scheduling over a line topology. More specifically, we design a minimal schedule on a line under which the queueing delay of a packet is at most T initially plus a total of at most $A_{\max} = T \cdot R_{\max}$ once the packet starts moving. Given that the links that support the routing as a result of Theorem 4 form a tree, we show below that a technique known as caterpillar decomposition allows us to partition this tree into a collection of lines so that the routing path of each connection goes through at most $2 \log n$ lines. Every time a packet switches to a new line, in the worst case, it pays for an initial queueing delay.

Knowing how to schedule on a line, we partition the routing tree into a collection of lines in a fashion similar to the *caterpillar decomposition* [7].

Lemma 5 ([7]). *Any tree can be partitioned into lines, such that the unique path between any two nodes traverses at most $2 \log n$ lines, where n is the number of nodes in the tree.*

Once the tree is decomposed into lines as in Lemma 5, the minimal schedule of Lemma 1 is used in each line.

Lemma 6. *The expected end-to-end delay of connection i is $4T \log n + \tilde{O}(k_i \log n)$, where k_i is the shortest path distance over the original network. Further, the schedule is minimal.*

Proof: The stretch of the spanning tree implies the length of the routing path in the selected tree is $\tilde{O}(\log n)$ times the shortest path distance in the original graph. Therefore,

the transmission time is $\tilde{O}(\log n)$ times k_i. From caterpillar decomposition, a routing path in the tree may be partitioned into $2\log n$ lines. Therefore, the queueing delay is at most $2\log n$ times the maximum delay in one line, $T + A_{\max} \leq 2T$. ∎

Observe that as $R_e \ll 1$, the length of the non-active period in each frame is $T - A_e = \Omega(T)$. Then, a packet that arrives at the beginning of this period has to wait $\Omega(T)$ before moving. Hence, the packet delay for connection i is lower bounded as $\Omega(T + k_i)$. Combining this with Theorem 4 and Lemma 6 we have,

Theorem 7. *The combined routing and scheduling scheme ensures an $\tilde{O}(\log n)$ approximation for delay minimization and $\tilde{O}(\log n)$ for energy minimization.*

V. Scheduling with Given Routes

In the previous section on the combined routing and scheduling scheme, we took advantage of the fact that the routes form a tree. In this section we focus on scheduling assuming routes are given as input, and assume these routes form an arbitrary topology.

A. Energy-Optimal Algorithm

We begin with a simple algorithm to demonstrate that a minimal schedule is always possible in a network of arbitrary topology, but a packet may experience a delay of T per link. Let k_i be the hop count of route P_i for connection i.

Theorem 8. *For a network with an arbitrary topology, the end-to-end delay of connection i is bounded by $T \cdot k_i$ under a minimal schedule.*

Proof: We first define a minimal schedule. For each time frame, link e is activated for the first $T \cdot R_e$ time steps. The schedule works as follows. At (the start of) time step jT, let S_e be the set of packets queueing at link e. During the time frame $[jT, (j+1)T)$, only packets from S_e advance along link e. That is, packets that arrive during $[jT, (j+1)T)$ have to wait till the next frame $[(j+1)T, (j+2)T)$ even if there is room during the active period $[jT, jT + TR_e)$. In the following we show that $|S_e| \leq TR_e$. We assume inductively that so far it takes one frame for each packet to advance one link. Therefore, S_e consists of packets from connections that start at link e and are injected during $[(j-1)T, jT)$, and from connections that have e as the ith link and are injected during $[(j-i)T, (j-i+1)T)$. The total rate of these connections is R_e. Therefore, $|S_e| \leq T \cdot R_e$. Since all packets from S_e are queueing at link e at the beginning of a time frame, they can be transmitted along link e during the active period of the frame which is the first $T \cdot R_e$ time steps. ∎

This minimal schedule can be seen as a direct application of Lemma 1 on the trivial decomposition of connection routes into lines in which each link is a different line. For that reason a packet experiences a line switch (and hence a delay of at most T) at each edge. Clearly, any other decomposition of the connection paths into lines would improve the end-to-end delay by reducing line switches. Unfortunately, there is not a lot of room for improvement in the worst case, since it is possible to find networks and connection sets such that, independent of how the decomposition is done, almost every connection i will experience $\Omega(k_i)$ line switches. However, experimental results presented in Section VI show that at a practical level this could be a promising approach.

B. Delay-Improved Algorithm

We show now that, by allowing the active period to be non-minimal, the maximum delay suffered by the packets can be reduced to $2T$, for large enough T. Due to space limitation we state without proving the following result.

Theorem 9. *For sufficiently large frame length T, all packets injected during one frame can reach their destinations during the active period of the next frame, and the length of the activation period is at most $T + O(k_{\max} \ln(mT))$, where m is the number of links in the network and $k_{\max} = \max_i k_i$.*

VI. Experimental Results

In this section, we compare via simulation two frame-based schedules, a coordinated schedule and a simple schedule without coordination. Both schedules are minimal, and hence are optimal in terms of energy. Therefore, we use the (average and maximum) delay of packets when using each of them as the goodness metric. The coordinated schedule, called here *schedule with coordination* (SWC) is defined for lines. It activates each edge e for a time A_e in each frame (with the active periods shifted along the line), and applies the farthest-to-go (FTG) scheduling protocol in each of the edges of the line, as was described in Lemma 1. The *schedule without coordination* (SWOC) simply activates each edge e for a time A_e at the beginning of the frame, and uses FIFO to schedule packets. This schedule can be seen as a greedy implementation of the schedule described in Section V-A.

Two sets of simulations are performed. The first uses a network which is a line, while the second uses a general network. In each scenario, the number of connections is fixed, with the terminals of each connection chosen uniformly at random among all the nodes in the network. The connection rate is fixed to $\rho = 1/50$, so that ρT packets arrive for each connection in each T-frame. Every packet arrives at a uniformly chosen time step within the frame. However, following the lines of Lemma 1, SWC does not schedule packets to move until the next frame (even if the link is active and there are available time slots). For any given set of parameters, the experiment corresponding to these parameters is performed 10 times, and the observed results averaged.

A. Lines

We first present the simulation results on a network which is a line of 40 links. In this network 50 connections with path length 10 are randomly chosen. Then, simulations with time frame size T in the range 50 to 250 are performed. Figure 1 shows the average and the maximum end-to-end delay observed in these simulations.

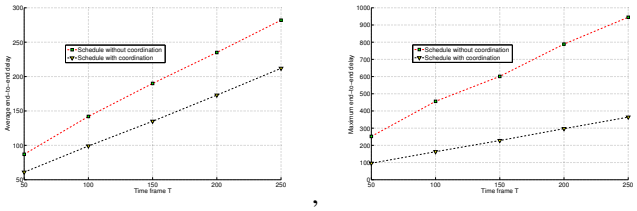

Fig. 1. Average (left) and maximum (right) end-to-end delay for a line network of length 40, with 50 connections of path length 10.

From Figure 1, it can be observed that the SWC provides significantly smaller end-to-end delay than the SWOC. Both the average and maximum end-to-end delays increase with T under SWC and SWOC. However, the growth is faster under SWOC, which means that the larger the frame, the more convenient it is to use SWC instead of SWOC. We have also observed that changing the length of the connection path does not change the fact that SWC performs better than SWOC. For instance, for path length of 30, SWC gives at least 26.8% better average end-to-end delay.

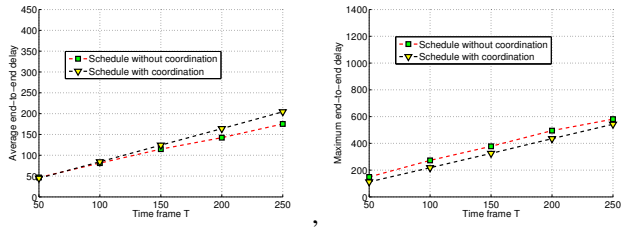

Fig. 2. Average (left) and maximum (right) end-to-end delay for $l = 1$.

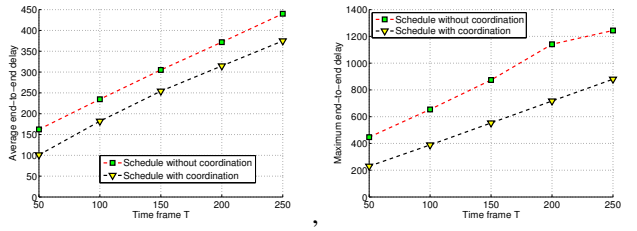

Fig. 3. Average (right) and maximum (right) end-to-end delay for $l = 10$.

B. NSF Network

The second network we consider is the NSF network, which consists of 14 nodes and 20 links. In this (and any arbitrary) network, SWOC works as it did in the line, simply using shortest path routing. However, we need to define how to make SWC work in the NFS network (and in any arbitrary network in general). The solution is to do line decomposition: decomposing the network into edge-disjoint lines, route connections along the lines, and use SWC in each line. In this case, we have manually decomposed the NSF network into four link-disjoint lines. For space restriction we do not show the decomposition.

When needed, a packet has to switch lines. In our simulations, when this occurs, the switching packet p is treated in the new line as any new arriving packet (and, in particular, p will not move until the next frame). This highly penalizes SWC at each line switch. We are interested in the impact of line switches on the end-to-end delay with respect to the path length, captured by a parameter κ, which is the ratio of the length of a connection path to the number of line switches. Since the original NSF network is small, we extend the NSF network by replacing each link with a path of length l. This allows increasing the ratio κ just defined without changing the number of line switches. Hence, in our simulations l roughly represents the above ratio.

In each experiment in the NSF network 100 connections are created. In each connection the same number of packets ρT are injected in each terminal node towards the other in each T-frame. As said above, these packets are routed via shortest paths when using SWOC and via decomposition lines when using SWC.

The results of the simulations are presented in Figures 2 and 3. For small l (e.g. $l = 1$ in Figure 2) SWC and SWOC have comparable performances. However, as l becomes larger (e.g. $l = 10$ in Figure 3), SWC provides significantly better delays than SWOC, both on average and in the worse case. Table I summarizes several parameters observed in the experiments for the values $l = 1, 5, 10$. It is easy to verify in that table

l	# of nodes	R_e Average	R_e Max	Connect. length Average	Connect. length Max	Line switches Average	Line switches Max
1	14	0.20	0.46	2.0	4.0	0.63	2.0
10	194	0.21	0.42	20.7	44	1.1	4

TABLE I
THE LENGTH OF CONNECTION PATHS AND THE NUMBER OF LINE SWITCHES.

that l in fact behaves as the ratio κ. Hence, from this table and the previous figures one can conclude that as the ratio κ grows (which is $2.0/0.63 < 10.5/1.17 < 20.7/1.1$, see Table I), SWC gives much better end-to-end delay performance than SWOC.

REFERENCES

[1] In *Proceedings of the Vision and Roadmap Workshop on Routing Telecom and Data Centers Toward Efficient Energy Use*, October 2008.

[2] I. Abaraham, Y. Bartal, and O. Neiman. Nearly right low stretch spanning trees. In *Proc. IEEE FOCS*, 2008.

[3] B. Awerbuch and Y. Azar. Buy-at-bulk network design. In *Proc. IEEE FOCS*, 1997.

[4] Y. Bartal. On approximating arbitrary metrics by tree metrics. In *Proc. ACM STOC*, 1998.

[5] P. Fishburn. *Interval orders and interval graphs*. Wiley and Sons, New York, 1985.

[6] Robert Hays. Active/idle toggling with low-power idle. IEEE P802.3az Energy Efficient Ethernet Task Force meeting, January 2008. http://www.ieee802.org/3/az/public/jan08/hays_01_0108.pdf.

[7] J. Matoušek. On embedding trees into uniformly convex banach spaces. *Israel Journal of Mathematics*, 114:221 – 237, 1999.

[8] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In Jon Crowcroft and Michael Dahlin, editors, *NSDI*, pages 323–336. USENIX Association, 2008.

[9] Gavin Parnaby and George Zimmerman. 10gbase-t active / low-power idle toggling. IEEE P802.3az Energy Efficient Ethernet Task Force meeting, January 2008. http://www.ieee802.org/3/az/public/jan08/parnaby_01_0108.pdf.