Using Low-power Modes for Energy Conservation in Ethernet LANs

Maruti Gupta
Department of Computer Science
Portland State University
Portland, OR 97207
Email: mgupta@cs.pdx.edu

Suresh Singh
Department of Computer Science
Portland State University
Portland, OR 97207
Email: singh@cs.pdx.edu

***Abstract*—Most Ethernet interfaces available for deployment in switches and hosts today can operate in a variety of different low power modes. However, currently these modes have very limited usage models. They do not take advantage of periods of inactivity, when the links remain idle or under-utilized. In this study, we propose methods that allow for detection of such periods to obtain energy savings with little impact on loss or delay. We evaluate our methods on a wide range of real-time traffic traces collected at a high-speed backbone switch within our campus LAN. Our results show that Ethernet interfaces at both ends can be put in extremely low power modes anywhere from 40%-98% of the time observed. In addition, we found that approximately 37% of interfaces studied (on the same switch) can be put in low power modes simultaneously which opens the potential for further energy savings in the switching fabric within the switch.**

I. Introduction

Recently there has been a growing interest in reducing the energy consumed in Internet devices such as switches, routers and servers. This interest is driven by the goals of reducing cooling costs in wiring closets in data centers, overcoming the problem of excessive heat dissipation as devices become smaller to enable faster speeds of operation and finally, limiting increasing energy costs.

In this paper, we propose algorithms for conserving energy consumption in Ethernet LAN switches and the hosts attached to them by shutting down the complex transceiver circuitry depending upon traffic arrivals, buffer occupancy and a bounded maximum delay. While there have been no studies that profile the energy consumption of an Ethernet switch, using available data we estimate that Ethernet interfaces can consume up to 20% of total switch power budget[1].

The idea of using low power modes in interfaces has been previously proposed in [4] and also explored by Christensen et al [5]. In the first case, the paper used a predictive algorithm to predict the arrival time of the next packet and made assumptions that an interface could transition from a low power mode to a fully operating mode on packet detection over the wire without losing any packets during the period of transition. However, such a transition from shutdown to operating mode is not practically achievable due to device physical constraints and packet loss is inevitable. In the second case, the approach is to operate the links at lower speeds during periods of low activity and thus reduce the energy consumed.

Our work here is different in that we attempt to shut down the link without making the assumption in [4] and we do not attempt to predict the arrival time of the next packet, which allows for a very small degree of error and is suitable only for the assumption made in that paper. Instead, we attempt to predict the number of packets that may arrive in a given interval of time that is large enough to allow the interface to shut the link down temporarily. The algorithm outlined in this paper uses technology available in current hardware and proposes small changes in the underlying link layer protocol which are easy to implement.

The remainder of the paper is organized as follows. Section II describes the current available technology, defines the problem and the algorithms we propose to solve it. Section III discusses our evaluation methodology and in section IV we obtain and analyze the results of our algorithms study the algorithms in depth using traffic traces.

II. Problem Definition and Approach

Currently, there are smart Ethernet transceivers available that automatically turn off after a few seconds when no power is detected on the other end of the link. Broadcom [2] as well as Intel [1] both have introduced network interface cards (NICs) that contain programmable sleep timers that allow the link to remain off for a certain time (typically these are programmed for 2.5 seconds to 5 seconds) when no power is detected and then periodically come back up on for a few milliseconds(ms) to check for power and then go back down, if there is none detected. Intel has also come up with NIC cards that automatically change link rate from 1Gbps to 100 Mbps or 10Mbps to save power in battery-operated devices. These mechanisms are not designed for cases when the link is operating normally or when it is idle or under-utilized.

In this paper we use the programmable capability of the sleep timer to dynamically change the operating power modes. In the scheme we describe in the next section, the interface

This work was funded by the NSF under award number NeTS-NR: 0435328.

[1]A Gb Ethernet transceiver can consume 1-2W depending upon whether it is copper or fiber[1], [2]. The Cisco switch 3750 under study has 24 gigabit copper-based ports and 4 fiber-optic ports. The total maximum operating cost of the switch is 190W[3].

Fig. 1. Simple model for a link.

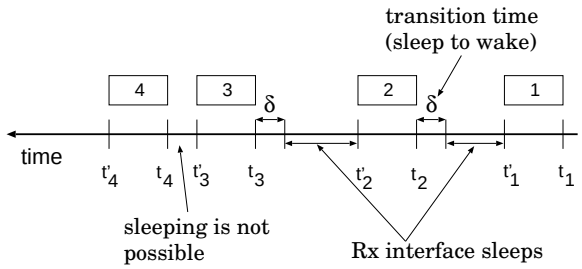

Fig. 2. Explanation of the fundamental problem.

either operates in shutdown mode or at the maximum negotiated speed. Figure 1 shows one half of a full-duplex link. We assume that when the link is off, it's off at both ends, both the Rx and Tx interfaces are asleep at each end and when the link is on, both the interfaces are powered on. We assume that packets queued for transmission at interface *a* when the link is off are buffered at *A*.

Figure 2 shows four packets that are sent from *a* to *b*. Packet 1 is the oldest and packet 4 the most recent. In the *optimal* algorithm, the link could be off for the interval $(t'_1, t_2 - \delta)$ between packets 1 and 2, and interval $(t'_2, t_3 - \delta)$ between packets 2 and 3. δ includes the time to transition between off to on state plus the link re-synchronization time. If $\delta > t'_3 - t_4$, then sleeping is not possible.

The key challenge here is in deciding *when* to turn a link off and for *how long*. The first algorithm we consider is called *On/Off-1*. In this algorithm, the upstream interface informs the downstream interface about when to sleep and for how long. We discuss how this communication can occur later in this section.

A. On/Off-1 algorithm

This algorithm runs at the upstream interface of a link. It is invoked whenever the buffer occupancy falls below a threshold (we use a value of 10%).

- B is the output buffer size at the upstream interface.
- w is the number of the most recent inter-arrival times. When a new packet arrives, the oldest inter-arrival time is discarded and the new inter-arrival time is added to the list
- λ is the mean inter-arrival time
- $\tau = \alpha B$ is the buffer occupancy threshold and $\alpha < 1$ (we use $\alpha = 0.1$ in our experiments)
- m is the number of packets in the buffer

1) If the link is on,
   a) if $m > \tau$, then do not sleep
   b) if $m \leq \tau$ then, check if link can be shutdown for t by estimating the number of packets n that will arrive in time t into the upstream interfaces' buffer. The goal is to ensure with a high probability that the total number of packets $n + m < \alpha B$.
   This computation is done by assuming that packet inter-arrival times are iid exponential random variables ([6] shows that inter-packet rates are piecewise Poisson). Thus, if $X_1, X_2, \ldots X_n$ are random variables for consecutive inter-packet times then $X = \sum X_i$ has a Gamma distribution. We find maximum t such that,
   $$P[X \geq t] \geq 0.9$$
   In other words, we find the maximum t such that the probability of more than n arrivals is less than 10%.
   c) if $t > \delta$ then the link is put in sleep mode for time $\min\{t - \delta, t^{max}\}$ where t^{max} is the maximum amount of time that the link can be put to sleep. The sleep time is transmitted to the downstream interface in an 802.3 frame.
2) If the downstream interface is asleep and the sleep time is set to expire, if $m = 0$ and $t > \delta$, then the upstream interface sends another 802.3 frame packet to the downstream interface to sleep for time $\min\{t - \delta, t^{max}\}$.

We use a value of $w = 5$ in the experiments due to a very high variability in the traffic traces. [2]

B. On/Off-2 algorithm

This algorithm is a modified version of the one above with two differences. We observed during initial runs with the On/Off-1 algorithm that both upstream and downstream interfaces wake up after sleep and then remain awake for some minimum time before going back to sleep. This behavior results in less energy savings during long idle periods. In the modified algorithm, the sleeping upstream interface does not wake up if the buffer is empty when the timer goes off. The downstream does wake up, it checks for energy on the link and if no energy is detected, it resets the sleep timer to the previous value and goes back to sleep. In addition, the upstream interface can wake up the sleeping downstream interface when its buffer grows beyond a certain threshold.

C. Link shutdown protocol

The exchange of information on when and how long to sleep can be done via 802.3 frames. Normally, when a link is re-established, an auto-negotiation protocol is run to, among other things, determine link type, link rate, synchronize clocks, etc. This process takes about 256ms and is too long for our algorithms since a large number of packets can arrive on a gigabit link in this interval. Instead, we propose that when a

[2]We also note that the behavior of this algorithm can be modeled as a single server queue where the server goes on vacation for t when the probability that the buffer will overflow is less than 10%. In this paper we do not develop a formalism for this model and is part of our future work.

link is turned on as a result of sleeping, the auto-negotiation not be run since apart from clock re-synchronization no other state change has occurred during t^{max}. With this simple addition, the algorithms can be seamlessly implemented in future deployments. Each of the two algorithms above require the upstream interface to transmit some information to the downstream interface. For the On/Off algorithms, the upstream interface sends $\log_2 t^{max}$ information per sleep. The data rate here can vary between $\log_2 t^{max}/\delta$ to $\log_2 t^{max}/t^{max}$.

III. Evaluation Methodology

The algorithms are evaluated in depth using traffic traces collected on our campus-wide LAN and described in section III-A. For our evaluation, the *metrics* and variable *parameters* used include the following. The metrics used are:

1) Total amount of time the link is off
2) Additional delay due to sleeping
3) Number of packets dropped
4) Additional buffer required to match the packet loss behavior when the interfaces do not sleep

Metric 4 is interesting because we explore the trade-off between power saved due to sleeping vs. extra power consumed due to additional memory used, if required.

The variable parameters we use in our study include the following:

1) Traffic traces collected at various ports of a high-speed switch.
2) Buffer size – In order to study this question, we ran the traffic traces for different buffer sizes at the upstream interface – 32, 64, 128, and 256 KB. We note that these buffer sizes are well within present day values used in both switches as well as host interfaces, [1], [2].
3) Maximum sleep interval t^{max} The maximum sleep times considered are: 1ms, 2.5ms, 5ms.

We used a fixed value of $\delta = 500\mu s$ for the sleep to wake transition time for the interfaces for all experiments. This value is based on proprietary transition time information for electrical circuits of the type used in gigabit NIC cards.

A. Traffic traces

We monitored a cross-section of ports on an on-campus backbone Cisco 3750 Catalyst stacked switch as described in Table I. This selection of ports reflects the diversity of LAN connectivity and thus gives us a comprehensive set of traffic traces to use in our study. The table also provides us with an overview of utilization of the switch. It is noteworthy that the switch is quite under-utilized, a fact that has been reported by researchers [7] in the context of internet backbone research.

We dumped packets using a port mirroring facility in the switch and to ensure no packet drops during logging, we monitored only one port at a time. For the ports connected to a router (into the DMZ) and VLAN trunk port, we only logged received packets due to the enormous amount of traffic flowing through the interface.

IV. Analysis of Experimental results

We tested the On/Off-1 and On/Off-2 on the traffic traces described previously. Since the traces were very diverse in terms of load and statistical properties, we expect our results to hold true for a wide range of possible traffic scenarios. [3]

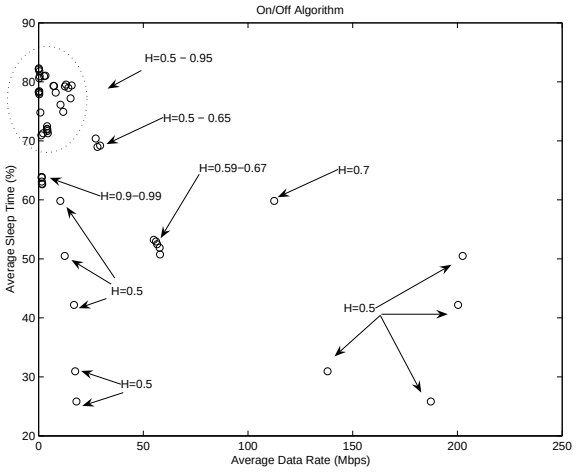

Fig. 3. Variation of Sleep times with Hurst parameter and data rate.

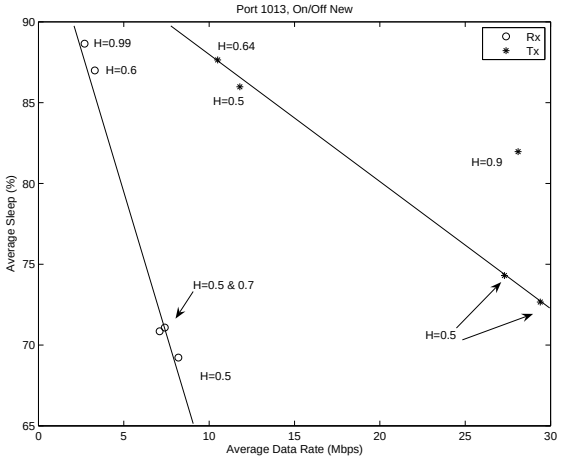

Fig. 4. Variation of Sleep times with data rate for 1013 (R^2 value for Rx fit is above 95% and is above 85% for Tx).

A. Relation of arrival process to sleep times

We analyzed the traces to explore the relationship between sleep times achievable, the traffic load and the statistical nature of the traffic (as characterized by H parameter for self-similar traffic). For this analysis, we split each individual trace into 100,000 packet blocks. For each block we measured the H parameter as well as the average load. *We only used those packet blocks that were self-similar*. The sleep algorithms were run on each of these blocks (we use a buffer size of 256KB and t^{max} = 2.5ms) and we recorded the average time spent

[3] A more comprehensive study with synthetic traces is beyond the scope of this paper.

Port	Description	Time logged	Total logged Pkts	RX	TX	Duration	Pkts/s	Mbps	%Util	Pkt Sz, Dist
1011	Solaris server	11:32am, 04/11/06	2425023	1312051	1112971	3.09hr	217.3	.75	.07	428B, bimodal
1015	Solaris server	11:51am, 04/12/06	724210	379739	344470	1.65hr	122.2	.47	.04	428B, bimodal
1020	Solaris server	1:30pm, 04/12/06	4209469	2534798	1674670	3.65hr	320	1.77	.17	694B, bimodal
2034	Linux server	4:45pm, 04/10/06	188681	100535	88145	0.12hr	543.3	3.5	.3	805B, bimodal
203	NFS fileserver	11:15am, 04/10/06	19990395	12259681	7730713	0.25hr	20312	111.63	11	687B, bimodal
1013	Mail server	3:45pm, 04/11/06	4996162	2805993	2190168	0.14hr	9865	29.8	2.98	378B, bimodal
1025	Vlan trunk port	4:46pm, 04/07/06	9282194			0.88hr	2905	12.8	1.2	551B, bimodal
1026	To router	11:52am, 04/13/06	1370225			0.03hr	11664	59.8	5.9	641B, bimodal

TABLE I
INDIVIDUAL PORT STATISTICS PER PORT, SWITCH3750, APRIL,2006

sleeping. We plot the sleep times versus load and mark each point with its H value for On/Off-1 in Figure 3. We see that the H value makes little difference in the sleep times. The only conclusion we can draw is that if the load is very low (around 0.1%) then for all H values we achieve very high sleep percentages (above 95%). However, this is not a surprising result. For higher load values, we can say little (indeed, similar plots for On/Off-2 and Optimal also show a similar lack of structure).

We next consider each port separately and look at the Rx and Tx data individually with the reasoning that individual servers running specific applications may have a more well-defined and consistent traffic pattern. As illustrated in Figure 4, we do indeed see a relationship. We find that there is a simple linear fit between sleep values and load regardless of the H value which ranges between 0.5 and 0.99 Similar patterns occur in other port traces as well. So a conclusion here is that the application type and load are the primary determinants of sleep times and not the Hurst parameter.

B. Variation in sleep due to buffer size

We studied the effect of four buffer sizes (32, 64, 128, 256 Kbyte) on the sleep times. Figure 5 shows this effect for ports 1011 and 203 representing the two extremes of loads in our traces. In both cases, as expected, sleep times increase as buffer size increases since more packets can be buffered during sleep. For port 1011 we also observe that there is a significant difference in sleep times between On/Off-1 and On/Off-2. This is because for On/Off-2 the interface sleeps uninterrupted during long idle periods without having to wake up and re-signal another sleep period and thus performs better than On/Off-1. For port 203, however, we see little difference in the performance. This is because port 203 had very few long idle periods. Finally, we see that for port 203, a minimum buffer size of 256KB is required before we see significant sleep times, which is not so for the much more lightly-loaded port 1011.

C. Effect of buffer size and t^{max} on delay

Figure 6 shows how average packet delay is affected by different buffer sizes (for a maximum sleep of 2.5ms). As expected, an increase in buffer size increases the queuing delay thus increasing the average delay. However, the delay is always less for On/Off-2, the reason being that for On/Off-2, the sleeping interface is woken up when the buffer occupancy

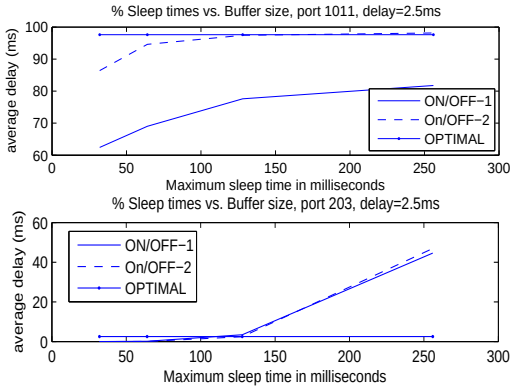

Fig. 5. Sleep time versus buffer size (1011,203), t^{max} = 2.5ms.

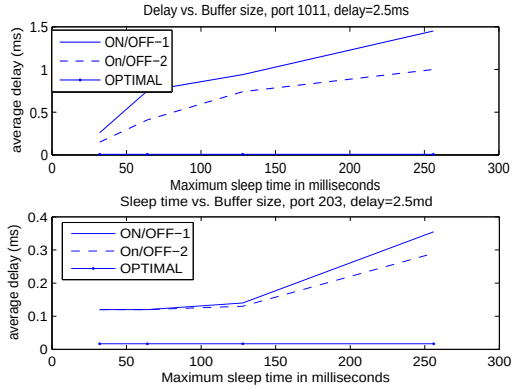

Fig. 6. Increase in delay versus buffer size (1011,203), t^{max} = 2.5ms.

crosses some threshold where as for On/Off-1, the interface only wakes up on expiry of its sleep interval. Thus On/Off-2 has a faster response to bursty periods and mis-predicted sleep times. The delay is noticeably different for ports 1011 and 203 and can be attributed to the difference in burstiness of their traffic. We discuss this in more detail later.

D. Packet loss and burstiness

We found that for buffer size greater than 32KB, most ports (including the more heavily loaded ones like 203, 1025, 1026) showed no packet loss for either algorithm. Interestingly, however, the lightly loaded ports like 1011 and 1013 showed some packet loss. The case of port 1011 is particularly interesting since it showed some of the largest peak burst sizes,

typically after long idle periods. This loss was observed for On/Off-1 at all buffer sizes, but not for On/Off-2 at buffer size 128KB, due to the faster response time of On/Off-2.

We examine the behavior of the algorithms during bursts by studying our two extreme cases, namely, port 203 and port 1011. For port 203, the mean packet arrival rate for every 5ms interval is 106 packets and the peak rate is 215pkts/5ms. Thus, the peak-to-mean ratio is approximately 2 for port 203. This allows the algorithm to better adapt to traffic, and thus we see no packet loss. For port 1011, the maximum rate is 169 packets and the mean is a mere 0.45 pkts, thus the mean-peak ratio is very high 369.4. In response to this burstiness, we also see highly fluctuating sleep times as seen in (Figure 8) ranging from 5ms to 0. We see packet loss occurs (for smaller buffer sizes) precisely at the point where we see small but high-intensity bursts.

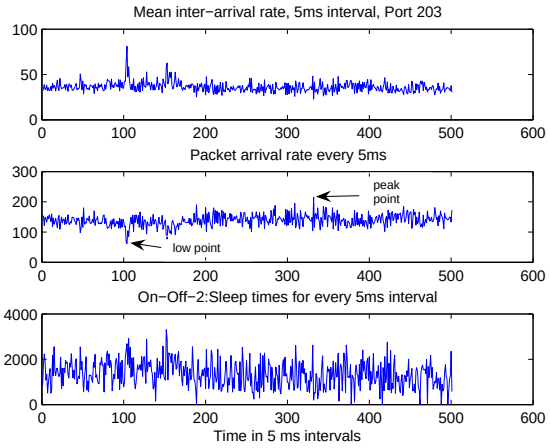

Fig. 7. Bursty behavior of port 203.

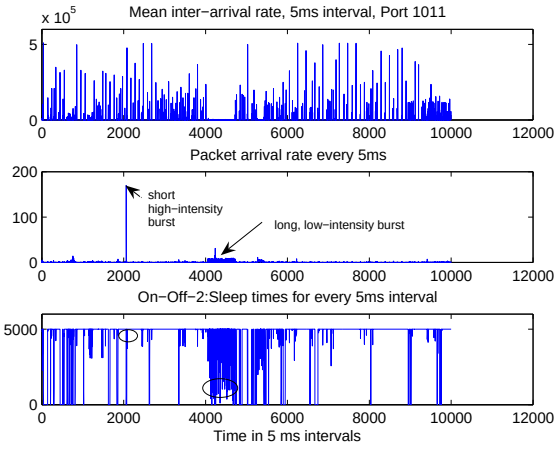

Fig. 8. Bursty behavior of port 1011.

E. Energy savings in the interface and fabric

We can estimate the *energy* savings possible at each switch interface by using a value of 0.9W difference between sleep and idle modes. Using the best sleep values from the results, we estimate that On/Off-1 can save between 40-90% energy and 33-80% and 28-84% for the On/Off-2 algorithm. Also, if we consider the Cisco 3750 where 20% of energy is consumed by the interfaces, we can reduce the energy cost of the entire switch by 10%. We also calculated the number of switch interfaces that are simultaneously turned off for an interval of 0.5ms for the 8 switch ports that are described here. The average number of interfaces that remained off for the *entire* duration of 0.5ms is 3 out of 8 or 37% (we are not counting ports that are off for part of the interval and thus the estimate we give is conservative). This means that about 37% of the time, other components of the switch such as the switch interconnect fabric can operate at 37% lower speeds and save energy either by shutting the internal links on/off as in [8] or by operating the links at lower frequencies, depending upon how they are designed.

V. Conclusions

We conclude that even for highly bursty traffic, it is possible to obtain significant energy savings through shutting the interfaces off. Our experiment results show that there is little relationship between the Hurst parameter and the amount of sleeping, but do show a relationship between load and sleep times. This needs to be validated further. We also found that for a given buffer size, packet loss is dependent on the peak to average burst size. Average load does not affect loss since the algorithm can adapt fairly well to high sustained loads, but can result in packet loss for highly bursty traffic. Additional delay due to sleeping is determined primarily by buffer size. For lightly loaded interfaces, delay is also affected by t^{max}, but not so for heavily loaded interfaces. Additional packet delays are small enough to not affect higher layer protocol behavior. We also observe that given that interfaces sleep, portions of the switching fabric within the switch can also be put to sleep.

References

[1] Intel, “Intel website.” [Online]. Available: http://www.intel.com/

[2] Broadcom, “Broadcom website.” [Online]. Available: http://www.broadcom.com/

[3] Cisco, “Cisco website.” [Online]. Available: http://www.cisco.com/

[4] M. Gupta, S. Grover, and S. Singh, “A feasibility study for power management in lan switches,” in *IEEE ICNP*, Berlin, Germany, October 5 - 8 2004.

[5] C. Gunaratne, K. Christensen, and S. Suen, “Ethernet adaptive link rate (alr): Analysis of a buffer threshold policy,” in *IEEE Globecom*, 2006.

[6] T. Karagiannis, M. Molle, and M. Faloutsos, “A nonstationary poisson view of internet traffic,” in *IEEE INFOCOM 2004*, 2004. [Online]. Available: ”citeseer.ist.psu.edu/696697.html”

[7] K. Xu, Z.-L. Zhang, and S. Bhattacharyya, “Profiling internet backbone traffic: behavior models and applications,” in *ACM SIGCOMM*, August 2005.

[8] L. Peh and V. Sorteiou, “Dynamic power management for power optimization of interconnection networks using on/off links,” in *11th Symposium of High Power Interconnects, 2003.*