Greening of the Internet

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

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

ABSTRACT

In this paper we examine the somewhat controversial subject of energy consumption of networking devices in the Internet, motivated by data collected by the U.S. Department of Commerce. We discuss the impact on network protocols of saving energy by putting network interfaces and other router & switch components to sleep. Using sample packet traces, we first show that it is indeed reasonable to do this and then we discuss the changes that may need to be made to current Internet protocols to support a more aggressive strategy for sleeping. Since this is a position paper, we do not present results but rather suggest interesting directions for core networking research. The impact of saving energy is huge, particularly in the developing world where energy is a precious resource whose scarcity hinders widespread Internet deployment.

Categories and Subject Descriptors

C.2.1 [**Network Architecture & Measurement**]: [Network Topology]; C.2.2 [**Network Protocols**]: [Routing Protocols]; C.2.6 [**Internetworking**]: [Routers, Standards]

General Terms

Algorithms, Measurement, Economics

Keywords

Energy, Internet, Protocols

1. INTRODUCTION

Recently, an opinion has been expressed in various quarters (see [5, 12]) that the energy consumption of the Internet is "too high" and that since this energy consumption can only grow as the Internet expands, this is a cause for concern. One may disagree, as we do, with the qualitative statement that the energy consumption of the Internet is too high, because it is a small fraction of the overall energy consumption. However, the absolute numbers do indicate a need to be more energy efficient. We use the analysis presented by these observers as a starting point to discuss an exciting new direction for future core networking research. We believe that if energy can be conserved by careful engineering then there is no reason why we should not do so as this has implications not only for reducing energy needs in the U.S. but also on speeding up Internet deployment and access in the developing world where energy is very scarce.

Device	*Approximate Number Deployed*	*Total AEC* TW-h
Hubs	93.5 Million	1.6 TW-h
LAN Switch	95,000	3.2 TW-h
WAN Switch	50,000	0.15 TW-h
Router	3,257	1.1 TW-h
Total		6.05 TW-h

Table 1: Breakdown of energy draw of various networking devices (TW-h refers to Tera-Watt hours and AEC to Annual Electricity Consumption).

Table 1 [14] summarizes the energy consumption by Internet devices in the U.S. as of the year 2000. These values are copied from Tables 5-59 (Hub), 5-61 (LAN switch), 5-62 (WAN switch), and 5-64 (Router) of [14]. The data is broken up based on network device type, which is useful in analyzing where and how energy savings can be garnered. In order to arrive at the various energy numbers in the table, the authors took into account the percentage of different types of devices deployed (e.g., number of CISCO 2500 type routers, number of 7505s, etc) and then used the average energy consumption values of these devices to arrive at the final numbers shown in the table[1]. Two energy values missing from the table are the energy cost of *cooling* the equipment and that of UPS (Uninterruptable Power Supplies) equipment[2]. The future expectation is that the energy consumption of networking devices will increase by 1 TW-h by 2005 [14].

Expressed as a percentage of total U.S. energy expenditure in the year 2000, the energy drawn by the devices in Table 1 accounts for approximately 0.07% of the total. Given that this is almost negligible in comparison to other energy

SIGCOMM'03, August 25–29, Karlsruhe, Germany.

[1]Note that the energy draw varies based on load and the values used in this study are based on observed average values.

[2]According to [14], air conditioning in data centers containing routing equipment costs approximately 20 – 60 Watts/ft^2.

costs, why should we care about energy conservation? There are three primary reasons:

- *Current Energy Inefficiencies:* If we use the total energy values from Table 1 (6.05 TW-h) and combine it with the estimated 20K – 35K Tera Bytes/Month total data routed on the Internet backbone in the year 2000 (from [2]), we obtain an energy cost of between 0.05 – 0.09 Joules/Byte. It is interesting to compare this energy cost against the *ideal cost* of wireless communication using 802.11b radios. Using values reported by one manufacturer (Lucent), it takes approximately 1.3 Watts to transmit and 0.9 Watts to receive. Assuming a 11Mbps link, it will take approximately 0.7 μs to transmit one Byte. Thus, the energy cost per Byte over a 100m link is 1.6 μJ. To equal the energy cost of transmitting one Byte over the Internet, we must transmit the Byte over a distance of between 3,200 km – 5,625 km (we divide the Internet energy cost by the wireless link energy cost to arrive at these numbers). If we assume that the Internet data refers to Bytes transmitted cross country (New York to San Diego is approximately 4480 km) then the wireless link is almost 1.25x times more efficient. We must add the caveat that this calculation does not account for idle energy costs or other processing costs at each hop. However, given that wireless networks are not known for their energy efficiency today, it is significant that the Internet costs appear to be in the same ballpark if not somewhat higher. Finally, we note that the wireless data above assumes an omni-directional transmission. If we assume that the transmitter uses a directional antenna (to bring it more in line with the point-to-point links of the Internet) then the difference is more glaring. For instance, assuming a directional transmitter alone (say with a $\pi/2$ beam), we ideally get a 8x improvement in range ($(4\pi)/(\pi/2)$) or an improvement of 10x over the wired Internet.

  We note that the high energy cost for the Internet comes about because networking devices expend a great deal of energy even when idle (they are powered on 24/7). Unlike monitors, or other computing equipment that satisfy *Energy Star* recommendations by going into various energy saving states when idle, networking equipment typically does not (there are no Energy Star recommendations for networking equipment). This is because maximizing network throughput and minimizing latency are the primary driving factors in network design[3].

  In terms of cost, 6 TW-h costs of the order of one billion dollars per year (at a cost of seventeen cents per kW-h, see [1]) and requires one nuclear reactor unit[4]. If we assume that the rest of the world should have Internet accessibility similar to that in the U.S., we will probably need approximately 6 billion/250 million $\times$ 6 TW-h = 144 TW-h of electricity using year 2000 standards (one may argue with this population-based extrapolation but we believe that the order of magnitude increase is correct). The energy needs will increase in the future as we replace older slower equipment with faster (and hence more energy hungry) newer equipment.

- *Enable Greater Deployment:* In many parts of the world, electricity is a scarce resource and this poses one of the barriers to widespread Internet deployment. In addition, frequent power outages reduce the uptime of the deployed Internet. If the energy consumption of the Internet devices is reduced, we can deploy more devices for the same energy cost and, given the same UPS capacity, have more of them up and running during periods of power outage thus improving overall network reliability.

  As an example, consider the total energy consumption in India in the year 2000 which was approximately 509 TW-h [3]. Using the population-based extrapolation for Internet connectivity, we see that networking equipment in India would use 6.05×1 Billion/250 Million = 24.2 TW-h or 4.75% of the total energy consumed. This is a significant fraction of the total (unlike the 0.07% for the US) and provides a strong motivation to make the networking equipment more efficient.

- *Benefits in the Event of a Disaster:* Networking equipment in a disaster-hit area will rely on their UPS batteries for operation. However, if we can have some form of low-power operating modes, these batteries will last longer. Thus, hospitals, police, and other agencies in the disaster-hit area will be able to access data stored in the affected area for longer. Furthermore, [9] discusses low-power operation for data centers (albeit at a degraded level of service). Combining low-power networking with low-power data center operation gives us an integrated solution for overall low-power remote data access in the event of a disaster.

Given the above reasons for conserving energy in the Internet, *we should consider how the architecture and protocols deployed in the Internet can be modified to meet the goal of energy efficiency.* Among the possible approaches that can be used are the following:

- At an individual switch or router level, we can put to sleep some of the subcomponents such as line cards when they are idle or clock the hardware at lower rates. Present day Internet hardware does not have this capability and we thus need to design future switch and router hardware to enable sleeping.

- At the network level, we can consider changing routes during low activity periods so as to aggregate traffic along a few routes only, while allowing devices on the idle routes to sleep. This requires changes to the manner in which layer 2 and layer 3 protocols work. Furthermore, sleeping may affect layer 4 protocols such as TCP and we need to study this impact and develop appropriate solutions.

- Finally, we can imagine modifying the Internet topology in a manner that allows route adaptation (via aggregation and sleeping) under a range of network loads.

[3]There are some exceptions, for example memory used in routers and some newer network processors do enter low-energy states when idle but, overall, the energy savings are still much smaller than what can be achieved.

[4]On average a nuclear power plant may have two reactors, each of which generates an average of 9 TW-h electricity per year, see [6].

> In other words, we advocate topologies such that there is a good correlation between the number of awake devices and load. As the load increases, more devices wake up and as load drops, more devices can sleep.

There has been a great deal of research on reducing energy consumption in ad hoc wireless networks and in sensor networks, and some of these solutions can be applied to our problem as well. Jones et al provide a good overview of these different techniques at various protocol layers [13]. One approach developed, which is relevant to our problem, is aimed at putting the radios into sleep states for as long as possible while ensuring network connectivity. Algorithms have been developed at both the MAC layer as well as at the network layer. At the MAC layer, approaches developed include distributed algorithms for sleeping where a radio goes to sleep after announcing this fact to its neighbors and the decision to sleep is based on the state of one's neighbors (such as S-MAC [16]). Other techniques include using TDMA or similar coordinated transmission scheduling (such as the Point Coordination Function in 802.11b) to allow radios to sleep when they can neither transmit nor receive. These algorithms, which determine when radios can be put to sleep, can possibly be used in the Internet context to determine when individual interfaces (and hence point-to-point links) can sleep.

At the network layer, the key approach used has been to perform route selection in a way that allows large numbers of nodes to sleep. An example of this approach is [10] where the algorithm exploits the high node density to put a large number of nodes to sleep while still maintaining network connectivity. The set of awake nodes changes over time to ensure a fair distribution of power usage. Mechanisms such as this can be used in our context to aggregate packets along few routes thus allowing many Internet devices to sleep during periods of low load.

To summarize, we note that sleeping appears to be the appropriate way in which we can maximize energy conservation in the Internet. However, in order to implement algorithms for sleeping, (1) the hardware of networking equipment needs to be redesigned to allow software-enabled sleeping, (2) routing protocols need to be modified so as to allow adaptation of energy consumption to load via aggregation and sleeping, (3) the Internet topology needs to be amended so that there are more options for route selection to allow sleeping and aggregation, and, (4) we need to study the impact of sleeping on protocols such as TCP with an eye on changing the protocol so as to adapt to the presence of sleeping nodes.

In section 2 we briefly describe typical router and switch architectures and then describe which subcomponents can be put to sleep and the impact on delay and loss rates. In section 3 we then describe the how and when of putting various subcomponents into sleep states. Finally, in section 4 we describe the impact of doing this on selected Internet protocols.

2. ENERGY SAVINGS OPPORTUNITIES

If we examine the wide variety of switches and routers available today (in this paper we limit our discussion to LAN switch and router equipment only although other devices in the Internet are also amenable to energy savings), we see an impressive diversity in their architectures. For illustrative purposes, let us use Cisco routers as architectural examples. The simplest routers (such as the Cisco 1700) typically have a shared memory architecture with a central CPU that makes all routing decisions. The interface cards are connected to the memory and CPU via a PCI bus. A step up in speed and complexity are the Cisco 7xxx families. Here again, we have a central processor that is responsible for routing tasks. However, the line cards are connected to the CPU and memory via two PCI buses and a TDM switch. Moving up in complexity is the Cisco 12000 router which includes a switching fabric connecting the line cards together, a central route processor that computes and distributes forwarding tables to the line cards, and line cards that have ASIC-based packet processing engines and memory buffers to store packets and distributed CEF (Cisco Express Forwarding) tables [7]. Finally, a relatively recent router family is the Cisco 10000 which uses a new switching technique called the Parallel Express Forwarding (PXF). PXF uses a parallel architecture consisting of multiple forwarding paths each containing an array of micro-coded network processors in a pipeline to handle different aspects of packet processing. As in the case of routers, there are a wide variety of LAN switches as well ranging in speed from 10/100 Mbps to Gbps and supporting anywhere from a few ports to 24 or more ports. These various switches display a similar increase in hardware complexity as in the case of routers.

In general, *to reduce energy consumption in a router or switch, we can envision putting some or all components of the device into low-energy sleep states or clocking the hardware slower.* Looking at the various architectures today, we can identify the following main components that can be put to sleep – memory, main processor, bus, line card processor/ASIC, and the switching fabric. Of these different components, we note that most memory today can enter various power saving modes. We therefore believe that using this form of memory will reduce energy consumption in networking devices. Of the remaining components, we believe that putting any or all of them into sleep (or clocking them slower) will result in significant power savings. Let us examine the feasibility of putting each of these components into sleep.

Line Cards: Line cards range in complexity from very simple interfaces to complex ones containing network processors, memory, interfaces, and switch fabric interface. Putting a line card or any of its components to sleep will save energy but, depending on how quickly they transition to the awake state, we may see significant packet loss and latency. If we assume that the line cards wake up automatically on sensing data on their input ports (this is a reasonable thing to do in hardware since the logic for sensing a transmission on a line is simple and consumes little energy) then, if it takes 10μs to transition to the awake state, on a line running at 1Gbps we can expect to lose approximately 10kbit of data and to potentially add a 10μs end-to-end delay. This type of behavior is clearly not acceptable and strategies to avoid it need to be developed. We can envision two possible ways to deal with this problem – a network-wide approach (we call this *coordinated sleeping*) in which the routing protocol aggregates traffic into few routes (during low load) to explicitly enable some interfaces to sleep or a link layer approach (we call this *uncoordinated sleeping*) where an interface sleeps based on local decisions alone. The link layer approach could work as follows – when an interface is about to sleep, it informs

its neighbor of the fact (on a point to point link this neighbor is well-defined, in a broadcast medium, all neighbors are informed via a broadcast). When the neighbor needs to send packets to this sleeping interface, it first sends a packet that forces the interface to wake up. Then, after waiting the necessary wake-up time, the actual packets are sent. This mechanism avoids the packet loss but still adds to the end-to-end delay.

The network processor on a line card can also enter power saving modes by being clocked slower when there is little network traffic. For instance, Intel has [4] announced a companion chipset based on the StrongARM processor to enable software dynamic voltage scaling (DVS) [15] for the IXP processor. Clearly, this form of technology will enable energy savings. The research challenge here is to determine when to slow the processor's clock.

Crossbar: Putting the crossbar into sleep in conjunction with line cards should not necessarily lead to any additional latency or loss. The reason is that when the line card is woken up, we can simultaneously wake up the crossbar as well. Thus, by the time packets start arriving at the line card, the crossbar is ready to route them. If, however, the line cards do not sleep but the crossbar does, then losses can occur at the input to the crossbar unless we dramatically increase the amount of buffers at the line cards.

Main Processor: The main processor is typically a RISC processor running at GHz rates. It may be possible to clock this processor slower when there is little network traffic. However, the question is when can this be done?

In summary, it appears that various subcomponents of the router and switch can be put into sleep and then powered on just before packets begin arriving (using the mechanism outlined earlier). However, this will, at the very least, result in increased end-to-end delay because of the delay in powering on the line cards and other components. In the remainder of this paper we focus on two issues – impact on Internet protocols of putting hardware components into sleep, and deciding when to put hardware into sleep states.

3. HOW AND WHEN TO SLEEP

When we think about putting line cards or the switching fabric or an entire switch or router to sleep, the immediate questions that arise are:

- For how long can these components sleep?

  When we wake up a sleeping device, there is usually a spike in the energy drawn. Thus, to be effective the device should sleep long enough to offset the additional energy drawn to wake up. Furthermore, given that a device takes x μs to wake up, the sleep period y must be longer than x. This clearly defines the *minimum* time for which a device can be profitably put to sleep. How long can it remain asleep? This depends on whether the sleeping is coordinated or uncoordinated. If the decision is uncoordinated, then we can see that the maximum sleep period will be dictated by the needs of periodic activity of Internet protocols – e.g., in the case of OSPF (Open Shortest Path First) the need to send and receive periodic Hello messages automatically puts an upper bound on this sleep interval. If, on the other hand, the sleep periods are coordinated network-wide (and if the Internet protocols are modified to support sleeping) then these periods can be longer.

- How is the decision to sleep taken?

  When a router or switch is making an uncoordinated sleep decision (i.e., in isolation), the best it can do is monitor traffic on all its interfaces, determine the inter-packet times and then put interfaces to sleep based on an estimate of the expected inter-arrival time. If this estimate is too long (i.e., a packet arrives while the interface is asleep), then the arriving packet wakes up the interface and the future sleep time will need to be recalculated. If, on the other hand, the sleep decisions are coordinated and made at the level of an area in an AS, then sleep times can be arrived at by estimating the traffic flow (again, based on some windowing measurement scheme), aggregating it, and then informing selected routers and switches to sleep.

- Which switches and routers are most amenable to sleeping?

  If we look at the Internet architecture, we note that depending on where they are located, some routers and switches are more amenable to sleeping than others. If the AS does not act as a transit AS for other networks then all the traffic is to/from the AS only. In these instances, it is easier to predict the traffic patterns and determine appropriate sleeping schedules. For instance, routers and switches within an area that route traffic from end users can probably sleep for long intervals during periods of relative inactivity (e.g., at night). Routers within the backbone of the AS can also sleep in periods of low load if there are multiple routes between them. It is possible to also put the border routers of stub ASes to sleep because they may not see much traffic (unless the AS provides web-based applications, in which case there may be constant traffic).

  In order to extend sleeping to Transit ASes, we believe that these ASes need to add additional links internally to enable better traffic aggregation during periods of low-load thus enabling large portions of these ASes to sleep. This will benefit ISPs (Internet Service Providers) because cooling costs tend to be high and sleeping will reduce the overall heat dissipation (in addition to reduced electricity consumption by the routers).

3.1 Examples Supporting Sleeping

In order to test the feasibility of sleeping, we collected sample traffic traces within our own AS. Figure 1 shows the portion of our AS that is relevant. Routers F and G are border routers with three interfaces facing out to the Internet and one interface each facing inward. Routers D and N sit on the boundary of Areas 1 and 0. We collected traffic for the gigabit interface at routers D and N facing into Area 1. We also collected data for the gigabit interface of these routers that connect into Area 0. For routers F and G we collected data at the interface into Area 0. Finally, we collected data at the interface of switch S that represents aggregate data from 20 hosts.

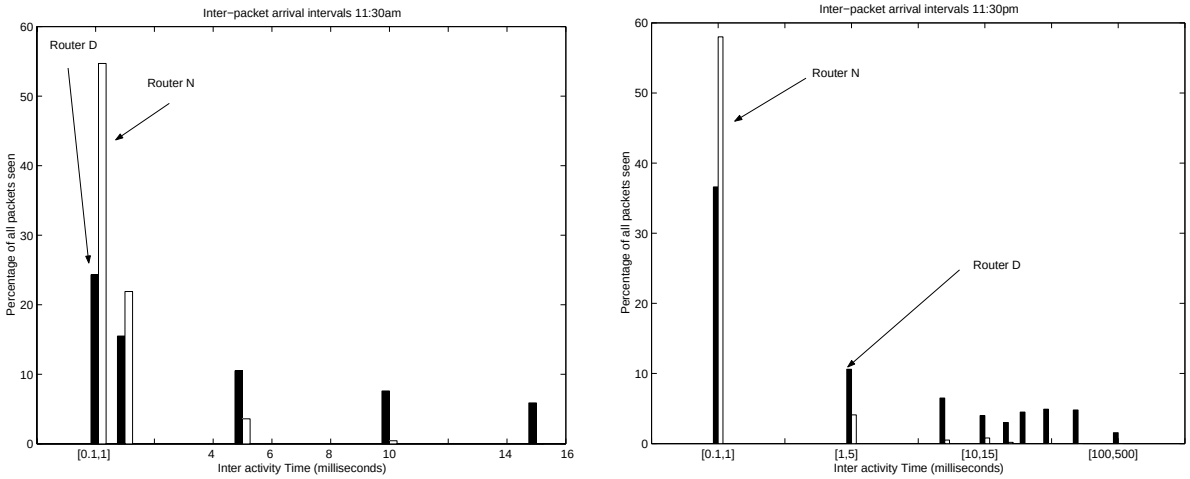

Figure 2: Histogram of inter-arrival times at an interface for area routers D and N.

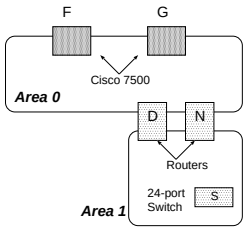

Figure 1: Data collection network.

Uncoordinated Sleeping

To determine if uncoordinated sleeping was feasible, we looked at traffic traces at the interfaces of routers D and N that face Area 1. The data was collected at 11:30am on a monday and 11:30pm that same night. For the 11:30am data, we counted the number of inter-arrival times that lay in the intervals [0.1 – 1ms], [1 – 5ms], [5 – 10ms], [10 – 15ms], and [15 – 20ms]. For the 11:30pm data, we added buckets [20 – 30ms], [30 – 50ms], [50 – 100ms], and [100 – 500ms]. Figure 2 shows the percentage of inter-arrival times that lie in each of these buckets for each of the two routers. The first thing to note is that for router N there is little difference between the morning and the evening trace, unlike router D.

How can we interpret this data as it relates to energy savings? If we assume that, to be beneficial, an interface should sleep for at least 1ms, then we note that for router D, 38% of inter-arrival times are over 1ms at 11:30 am and 40% are over 1ms at 11:30pm. These numbers are 26% and 5% respectively for router N. If we focus on router D, and assume optimal sleeping (i.e., the interface knows when to go to sleep and when to wake up) then we note that router D could sleep for as much as 95% of the time at 11:30am. We get this value by doing a simple calculation (which ignores packet processing times which are about 8μs, time to wake up, etc.) as follows. Let $t_i = 0.05, 0.55, 3, 7.5, 12.5, 17.5$ms denote the average values of the inter-arrival time buckets. Let p_i denote the probability that an inter-arrival time will fall in a given bucket (this is the data plotted in Figure 2) then, the maximum fraction of time the interface can sleep profitably is,

$$\frac{\sum_{[1--5ms]}^{[15--20ms]} t_i p_i}{\sum_{[0--0.1ms]}^{[15--20ms]} t_i p_i} = 0.9$$

Clearly this is a huge amount of savings and is a powerful argument for exploring this form of energy savings.

Figure 3 plots a histogram for inter-packet (or inter-activity) times on the interface of switch S that connects to an upstream gigabit switch. We see that over 90% of inter-packet times are 200ms or greater indicating the potential to power this interface off for extended periods.

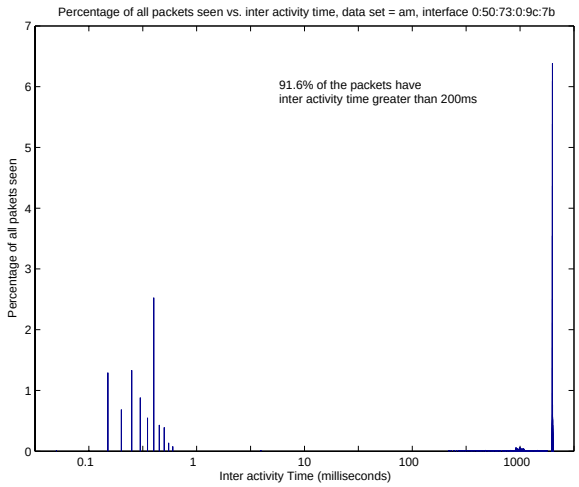

Figure 3: Inter-packet (inter-activity) times at interface of switch S (based on 2 hour traces in the morning on a weekday).

Coordinated Sleeping

Unlike uncoordinated sleeping, in coordinated sleeping the routers collectively decide which interfaces to put to sleep. To examine if this scheme would yield feasible results, we collected data on the inward facing interfaces of routers F and G and on the outward facing interfaces of routers D and

N. Figure 4 shows a twenty four hour plot of the average traffic (Feb 3 – 4, 2003) on the interface of F and G into our AS. We note that the primary router being used is router F (router G only shows periodic activity) thus it makes sense to put the inward facing interface of router G to sleep for extended periods. Figure 5 shows the aggregate traffic from Area 1 being sent into Area 0 by router N (similar traces appear for router D as well). We note here that there is a big drop in traffic at night and thus it should be possible to power off D and send all traffic thru N (or vice versa). In fact, both these routers are under-utilized even during the day and we can consider aggregating all the traffic through one of them alone.

4. IMPACT ON INTERNET PROTOCOLS

Putting interfaces on switches or routers to sleep can have serious side-effects because of the manner in which various protocols work. In this subsection we describe the possible problems that may arise for selected protocols and approaches for fixing these problems.

4.1 Impact on Switches

Switches perform unicast and multicast forwarding using forwarding tables determined by adaptive learning algorithms and in the case of VLANs (Virtual LANs) by using administrator defined port groupings. The adaptive learning algorithms require switches to learn the outgoing interface for a MAC address by snooping the sender MAC address of packets. Likewise, multicast forwarding tables are constructed by snooping IGMP (Internet Group Management Protocol, RFC 2236) packet headers. The entries in these tables typically have an aging time field after which it is flushed from the table. In our our subnetwork, this value is 20 minutes, however, the default is 300 seconds (RFC 1493). Some interesting questions here include, if we put an interface on a switch to sleep what impact will that have on the unicast and multicast forwarding tables? Given that parameters such as the aging value are user configurable, how will these values affect the interaction between the sleeping algorithms and the protocols?

Switches maintain a spanning tree (IEEE 802.1d) in order to prevent broadcast loops (including ARP loops). The switches run a spanning tree protocol and spanning tree packets are sent every 2 seconds. Furthermore, any changes to topology (i.e., new links or links going down) causes a recomputation of the spanning tree which may take up to 30 – 60 seconds to converge. This is clearly a problem for our approach of putting interfaces to sleep because, if done naively, it will result in constant spanning tree recomputations (and probably zero data throughput). Thus, we believe that 802.1d needs to be modified to (1) allow interfaces to sleep without sparking spanning tree construction, and (2) disable the requirement for the periodic 2 second spanning tree packet exchange for sleeping interfaces. It will be a research task to determine the impact of this on the correctness of the spanning tree protocol itself.

Finally, some switches allow "channel bonding" where two or more interfaces appear to be one virtual interface. Thus, if we bond together two 1Gbps interfaces, the link is then a 2Gbps link (the switch multiplexes packets over the two interfaces). We need to examine the impact of sleeping on channel bonding.

4.2 Impact on Routers in an AS: Examples

As we noted earlier, there are two types of sleeping – coordinated and uncoordinated. In the case of uncoordinated sleeping, logically, there should be no effect on any of the routing protocols because the interface will wake up when it receives a packet. However, depending upon the mechanics of the protocol, there may be some undesirable side-effects. In this section, we describe some of these effects for two popular protocols: OSPF and IBGP. In the current implementation of the OSPF (Open Shortest Path First, RFC 2328) protocol, if a router puts an interface to sleep and informs its neighbor(s) on that interface's link, those routers will generate LSA (Link-State Advertisement) packets indicating a failed link. This will, in turn, trigger a flood of LSA updates followed by a recomputation of the Shortest Path First (SPF) algorithm by all routers. This behavior is clearly unnecessary and expensive. Thus, the OSPF implementation needs to be appropriately modified so as to not treat uncoordinated sleeping as link failure. The same holds true for all the other protocols as well.

Implementing coordinated sleeping, on the other hand, does require significant design changes to existing protocols and may result in unexpected side effects. At a high level, the idea of coordinated sleeping is that during low load times, the routing protocol identifies single routes rather than multiple routes (as is done in OSPF) between source-destination pairs. The selection is further guided by the need to minimize the total number of awake interfaces in the network (note that it makes sense to identify interfaces that can sleep rather than entire routers because this gives us more flexibility in route selection). This requirement translates to the following fundamental changes to OSPF:

1. During periods when coordinated sleeping is possible (let us call this *network power conserving mode*), the SPF algorithm needs to be replaced by an algorithm that *identifies the minimal number of links to guarantee all-to-all routing while satisfying the QoS needs of the supported flows.* Consider a simple example where we have five nodes connected to form a ring. If we use SPF to generate routes, all the five links will be used and no interface can sleep. However, under some circumstances, deleting one link and using the remaining four is sufficient to satisfy QoS needs. This allows us to put the two interfaces on the deleted link to sleep. Thus, in general, the new algorithm needs to identify a minimum spanning tree or a minimum spanning tree augmented with some additional links so as to minimize the total energy consumption network-wide while satisfying flow QoS.

2. A drawback of using single routes is the impact of link or router failure. Recovering from these types of failures in our model will require additional coordination to identify new links that need to be woken up. In order to do this, we need additional protocol complexity in OSPF.

3. The Hello messages need to be restricted to only the awake interfaces. Furthermore, on broadcast multiaccess networks, the selection of Designated and Backup Designated Routers may need to be changed when the network enters power conserving mode.

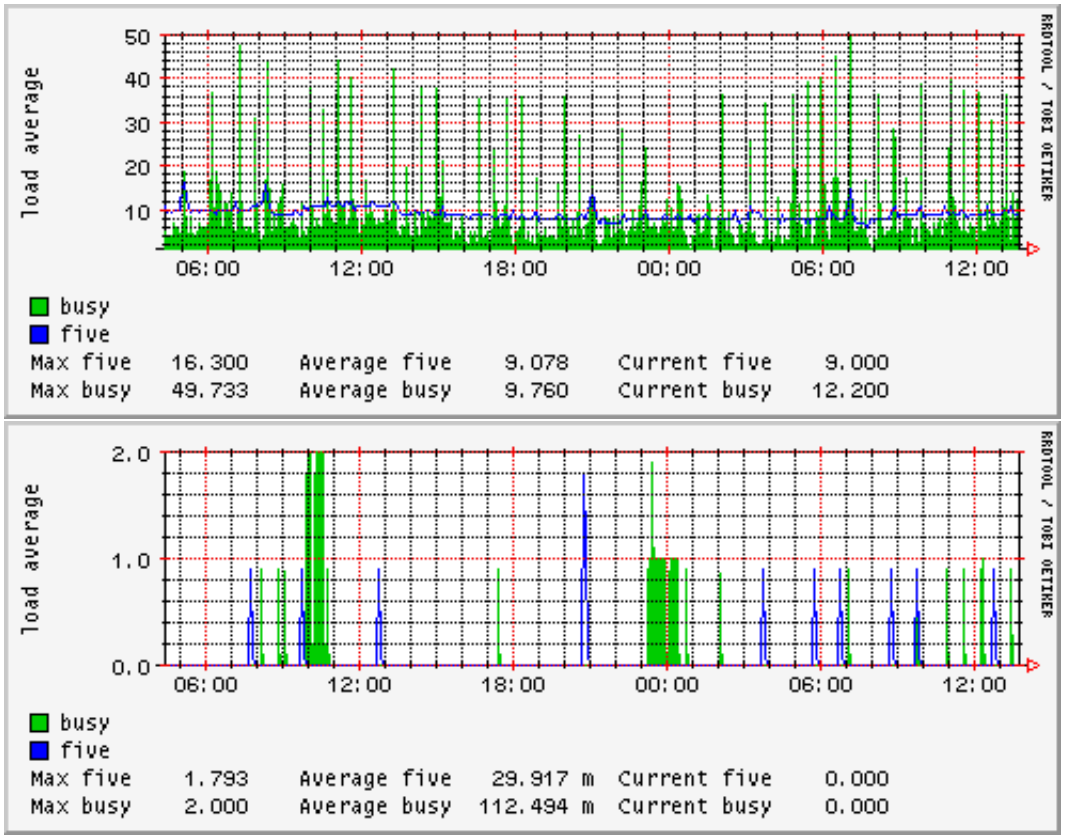

Figure 4: 5 minute avg for inward facing interfaces of routers F and G; *Load Average:* % busy time.

4. An underlying assumption above is that the routers somehow know that the load in the network is low and it is time to enter the power conserving mode. This is a non-trivial problem and we need to develop algorithms to predict this event and incorporate the algorithm into OSPF. We believe that overloading the Hello protocol to carry messages for this distributed agreement may be a good idea. However, this is another issue to be explored in detail because the long interval between Hello messages (as much as 10 seconds) can negate any short-term energy savings.

Putting interfaces or routers to sleep may be problematic for the IBGP protocol because of the manner in which the IBGP routers determine their most preferred route to external ASes. Several papers have described the problem of route oscillation, persistent forwarding loops, etc. because of the interplay between MED (Multi-Exit-Discriminator) and the cost of internal links when using route reflection and because of path asymmetry in IBGP [8, 11]. Putting interfaces to sleep means that several links in the network disappear. Thus, IBGP will need to be re-run to recompute the best routes to various external destinations.

This need to recompute routes in OSPF and IBGP points to two fundamental architectural changes to enable sleeping:

1. One can envision several power saving levels for the network. When all interfaces and devices are awake, we are at level 0. At higher power saving levels, a larger number (or percentage) of interfaces and devices are asleep. Since OSPF and IBGP need to re-run routing algorithms each time the system transitions to a different power saving level, it is important to ensure that every router sees the exact same network topology to prevent inconsistent and incorrect routes. Thus, all messages relating to sleeping need to be numbered appropriately.

2. There is a need for some form of centralized decision making to decide when to power off/on which interfaces and devices. This protocol would also be responsible for maintaining the current topology map that is then sent to all routers. In other words, this protocol (1) collects traffic data from the entire network to make powering-off decisions, and (2) maintains a database of the link state throughout the network and distributes this to all routers.

One final point to note is that the central decision making protocol must first *pre-compute the actual benefits of sleeping* prior to enforcing coordinated sleeping. This determination will require a pre-knowledge of energy consumption for protocol and packet processing at each router and each interface. Finally, we need to study the impact of sleeping on other protocols including EIGRP (Extended Interior Gateway Routing Protocol), ISIS (Intermediate System to Intermediate System), RIPv2 (Routing Internet Protocol), etc.

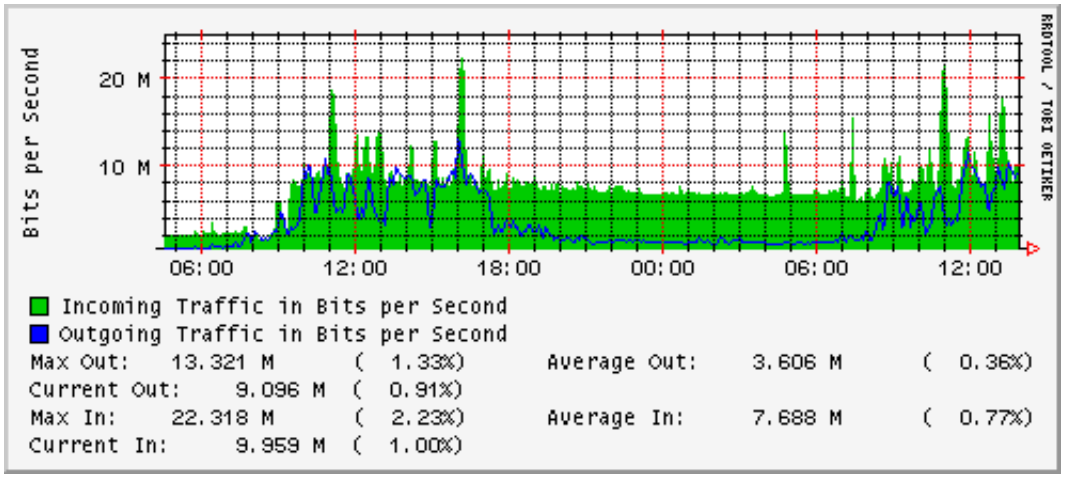

Figure 5: Aggregate traffic from router N into Area 0.

5. CONCLUSION

In this paper, we identify the problem of excessive energy consumption in the Internet and propose sleeping as the approach to save energy. We examine the impact of selectively putting interfaces to sleep on the implementation of switch protocols, and OSPF and IBGP routing protocols. It appears that sleeping is indeed a feasible strategy but it will require some changes to the current protocol specifications. Further, in order to maximize the amount of energy conservation, we note that some modifications to the Internet architecture may be needed (particularly adding more links to allow packet aggregation along fewer routes).

Acknowledgments

The authors would like to thank the anonymous referees as well as David Wetherall for detailed comments that have considerably improved the paper. David also provided the idea of using wireless energy costs as a point of comparison. The authors would also like to thank David Burns for assisting us in data collection.

6. REFERENCES

[1] Energy Calculator, http:// www.pge.com/ 003 _save _energy/ 003a_res/ index.shtml, (May 2003).

[2] http:// www.cs.columbia.edu/ hgs/ Internet/ traffic.html (Jan 2003).

[3] 2002 CIA World Factbook, http://www.cia.gov/cia/publications/factbook (2003).

[4] Intel IXP Companion Chipset, http:// appzone.intel.com/ pcadn/ product.asp? productid=133 (Jan 2003).

[5] International Energy Agency workshop on *The Future Impact of Information and Communication Technologies on the Energy System*, Feb 21 – 22, 2002, Paris, France. http://www.iea.org/weo/ict/agenda.htm.

[6] http://www.eia.doe.gov/cneaf/nuclear/page/nuc reactors/reactsum.html (Jan 2003).

[7] Vijay Bollapragada, Curtis Murphy, and Russ White, Inside Cisco IOS Software Architecture, Cisco Publications, 2000.

[8] Anindya Basu et al., "Route oscillations in I-BGP with Route Reflection", *Proceedings ACM SIGCOMM 2002*, Pittsburgh, PA¿

[9] J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. P. Doyle, "Managing energy and server resources in hosting centers", *Proceedings SOSP'01*, pp. 103 – 116.

[10] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "SPAN: An energy-efficient coordination algorithm for topology management in ad hoc wireless networks", *ACM MBICOM 2001*, July 26 – 21, 2001, Rome, Italy.

[11] Timothy G. Griffin and Gordon Wilfong, "On the correctness of IBGP configuration", *Proceedings ACM SIGCOMM 2002*, Pittsburgh, PA.

[12] Gilbert Held, "Emerging Technology: The Price of Power Consumption", *Network Magazine*, Sep. 5, 2001.

[13] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J-C. Chen, "A survey of energy efficient network protocols for wireless networks", *Wireless Networks*, Vol. 7(4), pp. 343–358, July 2001.

[14] Kurt W. Roth, Fred Goldstein, and Jonathan Kleinman, "Energy Consumption by Office and Telecommunications Equipment in Commercial Buildings Volume I: Energy Consumption Baseline", National Technical Information Service (NTIS), U.S. Department of Commerce, Springfield, VA 22161, NTIS Number: PB2002-101438.

[15] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy", *Proceedings 1st USENIX Symposium on Operating System Design and Implementation*, Nov. 1994, pp. 13 – 23.

[16] W. Ye, J. Heideman, and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks", *IEEE INFOCOM 2002*, New York, NY, June 23 – 27, 2002.