Greening the Internet; Power Optimization

Ashutosh Dhekne
Department of
Computer Science and Engineering,
IIT Bombay

Contents

List of Figures

Abstract

In this seminar, we shall discuss the need and techniques for power optimization of networking devices and protocols. We identify specific points in the networking world where considerable power wastage occurs and discuss them in detail. We discuss cost of various copy operations required to run TCP protocol and the energy expended in transmitting data using wireless communication. We identify the need to employ utilization based link speed variations in Ethernet. Various Internet protocols such as the OSPF protocol can be made energy efficient through appropriate modifications to allow devices to sleep when on low load. We discuss techniques to reduce the power consumption of over-provisioned servers. We understand that modifications to end systems are easier than modifications to networking protocols. Energy efficient protocols will be slowly phased in through initiatives from both the standards bodies and the manufacturers of networking equipment.

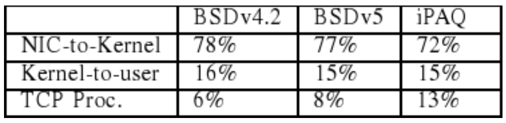
1 Introduction

The same topic means different things to different people. Power optimization means saving money to some; to others it means using the same set of batteries longer. Nevertheless, to all of us, it means a cleaner, healthier tomorrow.

Energy is not free. Its cost is two-fold. The first is the actual money we pay to buy energy. The second cost, which has only recently gained prominence, is the cost to the environment. Optimized usage of energy will lead to lowering of both these costs. We must be alert to tap all possible energy saving opportunities.

Building faster, smaller and more powerful computer systems has been a design goal in the computer industry for a long time now. However, optimization of power has been a consideration only in laptops and devices that are not mains powered. To have a greater power savings however, we also have to take into consideration systems that are AC powered. The total energy consumption by networking devices in the United States in 2003 was estimated to be about 6.05TW-h [1]. A recent publication [2] shows that servers in the United State contribute to about 0.6% of the total power consumption.

In this seminar, we investigate the energy saving opportunities in the networking world. We first discuss the power optimization possibilities at the client systems in Section 2. We observe that most of the cost of running TCP protocol comes from having to copy the data between buffers. A solution to this problem is the Zero Copy protocol which allows data to be copied directly between user space buffers and the NIC card. We then look at how wireless network cards can save power by sleeping intelligently while still guarantying bounded response time. In Section 3, we then discuss the energy saving avenues in the network. When Ethernet lines are working at low utilization, substantial energy savings can be achieved by switching to lower link speeds. Similarly, the Internet protocols may be modified to tolerate sleeping links without requiring a full recomputation of the routing paths. We then discuss briefly about energy optimization at the servers and recent works in that area in Section 4. Power conservation at the CPU is possible through dynamic voltage and frequency scaling and a few other techniques. Further, we look at the whole picture in Section 5 and the directions for future work.



	BSDv4.2	BSDv5	iPAQ
NIC-to-Kernel	78%	77%	72%
Kernel-to-user	16%	15%	15%
TCP Proc.	6%	8%	13%

Figure 1: Distribution of cost of TCP protocol

2 Energy Savings in Client Systems

The client systems that access the Internet require a fair amount of intelligence to run the Internet protocol stack. There are so many of these systems that finding mechanisms to save energy in them is assured to be most beneficial in terms of the total amount of energy saved. In this section, we shall study the cost of transferring data between the user application and the network card in Section 2.1 and then look at the savings possible in the process of transmitting this data over wireless in Section 2.2.

2.1 Cost of TCP

TCP accounts for about 90% of the total traffic on the Internet[3]. This ubiquity of the TCP protocol implies that a reduction in the energy consumption due to running TCP will benefit most systems connected to the internet. TCP consumes energy in the form of various copy operations, computation of checksums, maintenance of timers to implement the timeouts, triple dupacks and other book keeping activities. It also requires energy to send out packets on the physical layer, but all protocols would incur similar cost. The only cost of interest in the actual "send over physical layer" operation would be the overhead of using the TCP header rather than the UDP header.

The authors in [4] perform an analysis of the computational cost of TCP. They decompose the cost of running TCP into three primary categories: user-to-kernel copy, kernel-to-NIC copy and TCP processing cost. Whereas the cost of both the copy operations is same for senders and receivers, the TCP processing cost is different at the receiver and the sender.

It is observed that a significant chunk (72-78%) of the total energy is consumed by the kernel-NIC copy operation. The cost of actual packet processing is only about 6-13%. The checksum calculation cost amounts to about 30% of this packet processing cost on the test laptop used in [4]. The checksum calculation cost is a lower percentage (17-20%) of the processing cost for the iPAQ, which is also used in the measurements. The user-to-kernel space copy operation takes up about 15-16% of the total cost. These costs are summarized in Figure 1

The user space to kernel space copy cost can be eliminated using the zero copy mechanism. We shall discuss the zero copy mechanism in Section 2.1.2. We shall first look at techniques to reduce the cost of copying between the buffers and the NIC card.

2.1.1 Reducing copy cost

Two techniques are suggested for reducing the cost of the copy operation from the kernel space to the NIC. The first one is to maintain a buffer at the NIC that will serve any retransmissions required. Thus, data will be copied from the kernel to the NIC only once. However, it would require the NIC to know which packet to retransmit and would require as many separate buffers as the number of flows. In addition, there is no gain in doing this if retransmissions are very rare. The second technique is to copy larger chunks of data to the NIC. This, they claim, reduces the number of context switches and interrupts. The gain by using 1460-byte transfers instead of a small 40-byte packet, is about 2x for the laptop running FreeBSD and about 1.4x for the iPAQ. Though this method appears promising, it removes the natural coupling of a write command and packet boundary. When a single write copies, say, an entire sender's window, additional mechanisms are required for the NIC to be able to distinguish packet boundaries.

However, programmable NICs are available and protocol execution can be offloaded to the NIC itself. In that case, as suggested by [5], it is possible to avail of drastic improvements in the energy consumption by completely bypassing the operating system from the user to NIC path. Of course, it would require us to strip down all existing code and build new applications. This is an inappropriate proposition at this point of time. However, new applications may avail of this benefit if an increasing number of NICs (particularly the 1Gbps and above variants) support such functionality. This approach requires that the user lock the data pages in the main memory and inform the NIC of the physical address of the locked pages. The NIC can then pick up the data, perform framing, maintain timers and ensure that the host application will only see a reliable transport service running. [5] does not recreate the entire TCP protocol on the programmable NIC, but we can think of implementing it.

2.1.2 Zero Copy

In wake of the findings presented in [4], it is imperative that the cost of copying between buffers be reduced. The FreeBSD kernel includes a facility for eliminating data copies while reading from or writing to sockets. During normal network I/O, the CPU will not copy data at all. Data will be DMAed between the user's buffer and the NIC. During reads, data is copied to a user space buffer that will be directly presented to the user. During writes, the user provides the address of a buffer that is DMAes to the NIC. The only restrictions for sending data while using this functionality is that the data must be at least one page in size and must be page aligned. In addition, the application must not overwrite the buffer before the kernel frees the data and clears the copy-on-write mapping. Overwriting causes no gains to be realized since this will initiate copying as is normally done. The man pages for Zero Copy [6] mention that it is a good idea "...to set a socket buffer size appropriate for the application and network environment and then make sure you have sent out twice as much data as the socket buffer size before reusing a buffer. For TCP, the send and receive socket buffer sizes generally directly correspond to the TCP window size."

On the receiver side, the requirements are more stringent and impossible to meet without support from the NIC hardware. The NIC must support an MTU that is greater than the architecture page size. The packet payload must be at least a page in size and page aligned. In addition, the NIC must be able to split the packet into multiple buffers so that the payload starts in its own buffer. Header splitting, the mechanism that separates the header from the payload, may not be available in all cards. We may program this functionality if the

NIC at least supports splitting packets into multiple buffers. In this case, an optimistic approach is used and the NIC is programmed to split based upon the header size for the most common packet type. For other type of packets, the header or the payload may be fragmented into different buffers and will incur all the copy inefficiencies.

The DMA mechanism and zero copy is supported in many NIC cards. Software must make appropriate use of the functionality so that the energy cost of using TCP is lowered.

2.2 Wireless communication

In the previous section, we discussed about the cost of sending data from the application to the NIC. This is relevant for both wired as well as wireless communication. Substantial amount of energy is also required to transmit data from the NIC to the other side of the network. We now investigate the power conservation opportunities in wireless communication. Later, in Section 3.1, we shall investigate the power saving opportunities in Ethernet.

2.2.1 Enhancing the 802.11 PSM mechanism

Wireless devices usually operate in power-constrained settings. We may achieve longer battery life if we optimize these devices for power consumption. The 802.11 protocol has a feature known as Power Save Mode (PSM) in which a station is allowed to sleep between beacons if no data transfer is destined to it. When the station has some data to send, and is operating in the PSM mode, it returns to sleep after sending the data. [7] points out that this naive mechanism of sleeping immediately after a data transmission causes an unacceptable latency in receiving data. Most web traffic is TCP based. When a station issues a TCP SYN, it expects to get a SYN ACK from the server. If the station goes to sleep immediately after sending the TCP SYN packet, the returning SYN ACK will be buffered at the base station until the station wakes up again to collect it. The beacon period is typically 100ms. Hence, if the RTT from the station to the server is very small compared to the 100ms beacon interval, use of PSM causes an unnecessary slowdown of communication. An interesting situation may occur in which the total time taken for data transfer increases with increasing bandwidth between the station and the access point.

It is possible to impart an upper bound on the slowdown that will occur in data transfers due to sleep times. Stations typically wake up every beacon; however, we may configure them to wake up after skipping ListenInterval number of beacons. The protocol suggested in [7] uses this mechanism to modify the number of ListenIntervals the base station skips between wake ups. The slowdown can be bounded by a parameter p, which controls the rate at which the ListenInterval is increased. After a data transmission by the station, it stays awake for some amount of time (controlled by p) and then goes to sleep for one beacon period. If it finds that no data is scheduled for it even in the next beacon, it may decide, based on the value of p, to sleep for twice the number of beacons it slept last time. The mechanism for various values of p is depicted in Figure 2 which is reproduced from the original paper.

The station after transmitting a data packet always remains awake for that beacon period. If the station receives no data, after every 1/p number of bea-

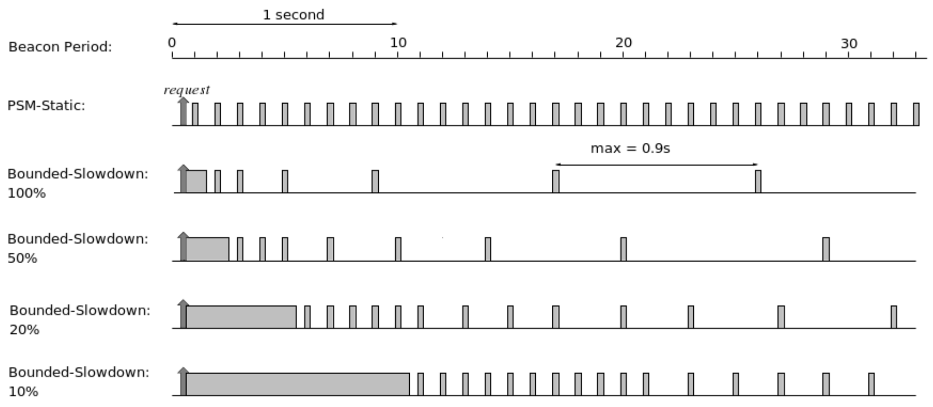


Figure 2: Working of the PSM mechanism and Bounded-Slowdown for various alternative bounds expressed as p.100 percent.

cons, it doubles the ListenInterval and progressively sleeps for more time. Thus, if p is set to 1, the ListenInterval is incremented at every beacon. Similarly, if p is set to 0.1, the station remains awake for 10 beacons and then sleeps between each beacon for the next 10 transmissions. Later, it sleeps for two beacons before waking up. In the next cycle, it shall sleep for 4 beacon periods. Thus, when the value of p is large, we are allowing a large slowdown to occur. Hence, the station reacts quickly to no data transfers being done and sleeps for increasingly more number of beacons in each round. When the value of p is small, the station reacts slowly and increases the time to sleep once every 1/p (expressed as a fraction) rounds. The designers have imparted a 0.9-second maximum sleep period to ensure that the higher layers do not time out.

Though this appears to be a good strategy, [7] does not experiment with more than one station. More number of stations each possibly working with the same protocol may cause unforeseen problems with the protocol. In addition, other questions like how much will the AP be loaded by adopting this procedure and would it require any firmware changes to the AP need to be answered. Implementing the protocol over the existing PSM mechanism should not be very difficult since the only new feature this protocol would require is to inform the AP that it would be sleeping for the next ListenInterval period.

2.2.2 Proxy Based Wakeup

A device that has a very low utilization can save a lot of power by turning itself off during periods of inactivity. We require a dynamic mechanism in which the device powers on when it is supposed to be utilized. A technique to accomplish this is to use a second device that will power on the main device when it detects relevant activity. This second device, also called a proxy, must consume much less power than the main device. [8] and [9] have employed a similar idea in two completely different settings to achieve similar goals. [8] uses a Telos mote as the proxy to wake up the main Soekris board in long distance wireless links. [9] have connected a low power radio to a PDA so that the main PDA does not have to keep its radio on all the time when it is not transmitting or receiving any data. They also require changes in the access point.

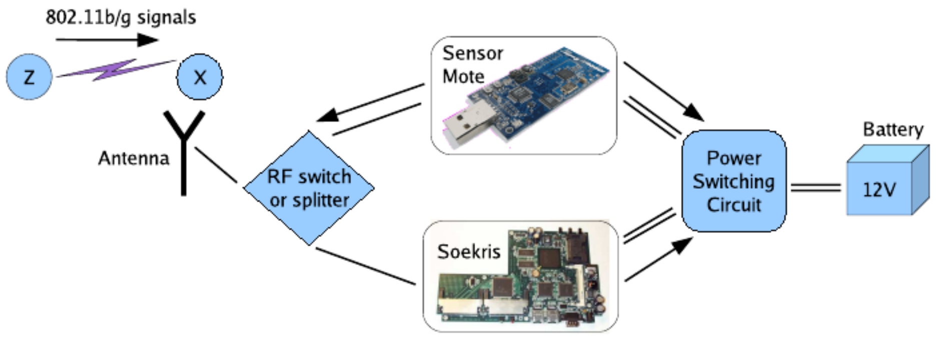


Figure 3: Wake-on-WLAN Architecture

In the Wake on WLAN [8] paper, the authors use a Telos Mote to wake up an 802.11 device. The general setup as shown in the paper is reproduced here in Figure 3. Long distance directional links are used to communicate between different Soekris boards in a wireless mesh network. Communication is not always active and there are substantially long times when no communication takes place. The authors have proposed a mechanism to put the main Soekris board to sleep during periods of inactivity, while continuing to monitor the air for any wireless activity using the Telos Mote. Since the mote can only detect an increase or decrease in the energy levels over a channel, it will wake up the main device whenever a data transmission occurs in the vicinity. Thus the solution, though elegant in the setting discussed in the paper, may not be suitable as a general solution for other situations. Some tricks are possible to make this approach more sophisticated. For example, a station may start with transmitting a sequence of bits that may be interpreted by the low power device. Two simple methods exist to encode the sequence in energy. The first one is to vary the width of an energy pulse and the second is to vary the energy level. This is required since the low power proxy may not be able to decipher the 802.11 modulated bits. Also, the latency that is acceptable in the setting discussed in [8] may not be acceptable in many other applications. To circumvent this problem, we may make use of different sleep modes available for most devices. For example, we may partially turn off a device by hibernating it so that time required to turn the device on is much lesser than turning it on from a completely off state. This way, similar to [7], we may be able to ensure an upper bound on the time taken for the main device since we know the time to wake from each sleep mode.

3 Energy Savings in the Network

After the client system transfers data to the physical communication link, it flows through the network before reaching the destination. Most of the Internet backbone is connected through wired links and hence we consider only the wired aspect of the network. A network comprises of the networking devices such as the routers and switches, the physical lines connecting them and the

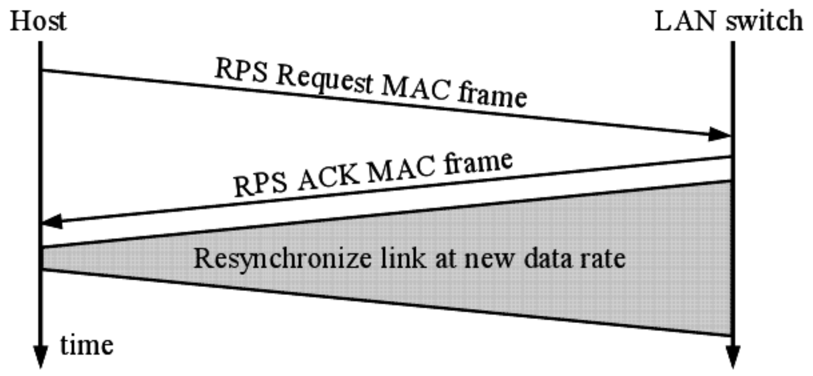


Figure 4: RPS handshake and link resynchronization

protocols used for data transfer. Within the premises of most corporate offices, universities and other establishments, computers are connected using Ethernet. Edge routers then connect these internal networks to the Internet. We first look at the energy saving opportunities in the Ethernet. Then, in Section 3.2 we investigate the power optimization avenues in the context of switches, routers and the protocols run on them.

3.1 Ethernet

There exist millions of Ethernet links from desktop PC to LAN switches across the world. Each of these PCs has at least one network interface card that is always powered whether or not the link is used. Further, it is always powered for the fastest possible link speed regardless of the link utilization. It is possible for faster links to operate at lower link rates and save significant power. One Gbps links consume about 2-4 W more energy than 100Mbps links. Investigations about techniques that may be employed to decide the link speed based on the utilization are underway and Rapid PHY Selection appears to be a promising technique.

In existing Ethernet protocol, when devices are connected through a network cable, they perform auto negotiation to decide the link rate and mode of operation (half duplex or full duplex). Auto negotiation makes it feasible to have devices supporting different link speeds on the same network. However, auto negotiation is typically done only on connecting a device anew. Therefore, the inherent latency of about 1 to 4 seconds is acceptable. Auto negotiation uses physical layer techniques since, just after connection, it is yet to set up any meaningful communication at the higher layers. Since this mechanism must support the slowest link speed available, auto negotiation works too slow for allowing dynamic switching of links.

We need a mechanism that allows us to switch the link speed quickly depending on the current load or utilization over that link. This new mechanism is called Rapid PHY Selection [10]. A special MAC layer frame is used for this purpose. The protocol, shown in Figure 4 borrowed from the original paper, includes an exchange of two frames to initiate a line rate change. The first one is a MAC control frame adapted from the existing PAUSE frame format. The second is a ACK/NACK frame. The modifications proposed are in the opcode field for identifying the frame as a MAC control or ACK/NACK frame and in the parameter field for indicating the link rate. It is proposed that Adaptive Link Rate (ALR) capability is advertised through auto negotiation during link setup. When a node wishes to switch to a lower link speed, it sends a MAC control frame with the desired lower link rate to the peer. It sets a timer associated with this request and if the timer expires before it gets either an ACK or a NACK from the other node, it retransmits the control frame. The sending of the initial MAC control frame is always done at the existing link rate. When a node receives a request to switch to a lower link speed, it stops all data transmissions and checks if it can switch to the lower link rate. If it decides to switch to the lower link rate, it sends an ACK and then switches to the lower link rate. The node must resynchronize its clocks at the new rate and continue to send data at the lower rate. If the receiver of the MAC control frame decides in the negative, it sends a NACK to the first node and does not alter its link rate. If a node requests switching to a higher link rate, the other node always replies in the affirmative and resynchronizes its clocks to the higher link rate.

The decision of switching between the link-speeds is taken by the node initiating the switch through mechanisms employed locally, possibly based on the recent utilization of the link. The authors of [11] have discussed two possible schemes. The first scheme is based on dual threshholds. If the output buffer queue-length exceeds a set threshold qHigh, the node initiates a request for using faster link rate. The other side always agrees to a switch to higher link rate. If the output queue size decreases below qLow, the node initiates a switch to lower link rate. The other side may disagree by sending a NACK frame if it has data output queue-length larger than the qLow threshold. A potential problem with this technique is that it may induce link rate flapping. This may occur if the data output rate is uniform such that it exceeds the qHigh threshold very often for the lower data link rate, but cannot keep the buffer over the qLow threshold for the higher data rate. In this case, the second strategy suggested in [11] is more useful. Two different thresholds are kept to measure the queue length and the actual number of bytes sent in an interval tUtil. A transition is made to the lower link rate only if the current queue length is lesser than the qLow threshold *and* the number of bytes sent in the last tUtil time interval is smaller than the uThresh threshold. Both these parameters can be analytically fixed for given link rates.

While RPS is being done, that is, while the physical channel is synchronizing to the new frequency, new data may arrive at one of the nodes. It cannot be put on the link before the resynchronization is complete. Thus buffering is required at the transmitting node. If the time required for RPS switching is less than the buffer capacity of the node, no data will be lost. If, however, the buffer is smaller than the amount of data generated data loss is bound to occur. The use of RPS may have interactions and undesired behavior when employed with higher layer protocols. For example, [10] says that UDP will never attempt to recover the packets lost due to RPS. TCP will eventually recover all lost packets but not without slightly increasing the latency of transmitting the data. We may need a PAUSE-like interface for the higher layers to stop sending data when the RPS is underway. An interesting side effect is that when the data rate at a particular link increases, and we wish to switch over to the faster link speed, we must first switch the link off for all data to drain out of the pipe and then resynchronize it to the higher link rate. We are thus stopping any data being delivered for a small amount of time, when, in fact, we want data to be delivered faster!

3.2 Switches and Routers

In order to save the energy consumed by switches and routers, we may put some or all components of these devices to sleep when the utilization is low [1]. Though these devices, depending upon their capacity, vary substantially in terms of configuration, they typically contain line cards, crossbar and a main processor. Each of these devices show different opportunities for power savings.

3.2.1 Opportunities at Component Level

Line Cards: We may put a line card to sleep if no traffic has been observed from a link and we expect a sustained lean period. However, since the time of arrival of the next packet is unknown, a sleeping device may cause both a data loss as well as increase in end-to-end latency. For example, if a 1Gbps card sleeps when it sees no traffic for some time, and requires 1μs to wake up, it will lose about 1kbps data when a packet does arrive and also add a 1μs delay to the transmission. If a packet hits multiple sleeping devices during its journey through the network, this delay may become significant. The packet loss can be avoided either through a network-wide approach in which nodes know which nodes are asleep at any particular time. Network protocols aggregate traffic to other routes to avoid disturbing these sleeping nodes. The synchronization that this type of protocol will require is likely to make it a poor choice. Another way to avoid packet loss is a link layer approach in which a node informs its neighbors that it is going to sleep. When the neighbor needs to send packet to a sleeping node, it shall first send a wakeup call to the sleeping node and only start transmission of the actual packet after waiting for the necessary wakeup time. This protocol becomes more and more difficult to implement if the line card has multiple sleep modes with differing wakeup times.

Crossbar: The crossbar refers to the interconnect circuit between the line cards and the main processor. The crossbar can be put to sleep along with the line cards and be woken up along with the line card. If crossbar sleeps independent of the line card, then the crossbar may itself become a source of increased latency. Packet loss can be avoided in this case using buffers at the line card. It is however not clear how much power is actually expended by the crossbar circuitry and whether the benefit exceeds the cost of having large buffers at the line cards.

Main Processor: The main processor may save substantial amounts of power by either clocking slower or through going into deeper sleep states when the load is low. We must note, however, that unlike line cards, the main processor is kept busy by more than one link. It is unlikely that all the links are under low load. Depending on the scheduling mechanism used on the processor, when fewer links are active, they may see a shorter response time because more CPU time will be available to each of them. If, however, the scheduler puts the CPU to sleep when on low load rather than giving more time to the active tasks, the response time may not reduce even when most of the lines are inactive.

3.2.2 Impact on Protocols

Switches: Switches are adaptive self-learning devices and store the knowledge of where ethernet frames are to be forwarded. Switches learn of the appropriate outgoing interface for a MAC address by snooping on the incoming MAC headers. For multicast traffic IGMP headers are used to achieve a similar effect. These stored entries in the switch's table age over time and need to be refreshed to avoid flushing. Thus, a switch that sleeps may end up having old entries in its tables or, if it decides to start anew, could spend enormous time learning the outgoing links all over again. Moreover, to prevent loops at the link layer, switches in a network are required to participate in the computation of a spanning tree. To account for failing links or devices, spanning tree packets are sent every 2 seconds by all participating switches. Missing these updates or defaulting on sending updates will result in recomputation of the spanning tree, which requires 30 to 60 seconds to converge. Thus, a switch going to sleep or waking up may disrupt the network. Modifications will be required to accommodate sleeping but alive switches and allow them to send spanning tree updates after longer intervals. Though it is a very interesting research problem, its wide acceptance is doubtful at this stage.

Routers: The OSPF protocol, which is used most commonly by routers, generates link state advertisement packets to keep all the routers informed of the current network topology. An interface that goes to sleep may cause other routers to incorrectly infer a failed link and trigger unnecessary LSA updates. Modifications to the widely accepted OSPF protocol are imperative if we wish to save energy by putting interfaces to sleep. However, just like switching algorithms, OSPF is so widely used that modifying it will be unpractical. Similar problems also exist for protocols like the IBGP, which determines which routers to use to communicate with external Autonomous Systems. Coordinated sleeping may be incorporated so that unnecessary updates are avoided in all these protocols. However, this coordination itself would require protocol modifications and a centralized control. The Internet works on the very pretext of distributed control and this approach of having a centralized sleep coordinator is bound to have its own issues of scalability and reliability. In summary, it is relatively much difficult to inculcate energy savings in the networking backbone than in the individual systems. Intelligence needs to be added to the Internet protocols such as OSPF so that the routers defer waking up sleeping nodes unnecessarily, but put to use the services of these interfaces when load exceeds a "threshold". It is like maintaining two different topology maps; one for actual connection topology and another one for currently awake paths (or links). Sleeping interfaces can be woken up when such need arises.

4 Energy Savings at Servers

By servers, we mean any computer system that provides data to other systems over a network. Proxies, data centers and web servers are all types of servers. Servers are generally built to support the maximum estimated load. However, compared to the maximum load at a server, the average load may be much less resulting in gross over-provisioning of the server. If each of the components of such a system operates at full speed, the system will be doing much less work during wee hours and still consuming the same amount of power. Intelligent techniques are required to determine the optimal clocking frequency so that the system is able to perform just with the expected speed and not waste any more energy than necessary.

The authors in [12] observed that the CPU power consumption shows maximum variations in power consumption related to the load. Since other components show less variability, they focus on the possible optimizations at the CPU. Power dissipation of a CPU is inversely proportional to the square of the voltage at which it operates and inversely proportional to the clock frequency. Using a simulator that estimates the energy consumption for given workloads, the authors have shown energy savings of 20%-35% when dynamic voltage and frequency scaling is used. The simulator examines the system load in the last time quantum and decides to step up the frequency by 33MHz if a high-load threshold is crossed. If the low-load threshold is crossed from above, the simulator steps down the frequency. Voltage is appropriately adjusted. In case the system was moderately loaded, with none of the thresholds being hit, the frequency and voltage is maintained at the current level. They have performed actual power consumption measurements by intercepting the power leads from the SMPS. More sophisticated techniques exist and the processor manufacturers have started making public data from such measurements. The white paper from AMD [13] is an example. Such controlled measurements allow the manufacturers to accurately measure the power consumption of different parts of the processor in addition to the power consumption of the processor as a whole.

Many techniques are employed to reduce the power consumption of modern computers [14], [15], [16]. These include dynamic voltage and frequency scaling (DVFS), clock throttling and selectively powering components that are needed. These dynamic techniques can be controlled through both software and hardware, but definitely require hardware support.

5 Discussion

The design of the Internet, its protocols and hardware, did not have special focus on power optimization. As a result, various areas have been identified which could do better in terms of energy efficiency if appropriately modified. Though this work divides the avenues of power optimization according to their place of application, a global picture is not difficult to visualize.

Client systems including PDAs, Laptops and Desktop PCs shall use power efficient methods to remain connected to the network. Wireless devices shall not continuously probe for access points. Systems would put their line cards to sleep when connections are not needed. Processors will clock no faster than required by the current application load. The network equipment will distribute its load to allow interfaces to sleep. The day-night pattern at different AS will be exploited to maximize sleep time for the routers in that area which is generating low load. Network protocols will be optimized to send out only minimal link alive packets possibly triggered by proxy low-end systems so that the main system is still sleeping while its route is kept alive. The servers will use dynamically changing CPU frequency and voltage and would clock slower when on lighter load. Lower power consumption will also cause lower heat dissipation reducing the cooling cost of data centers.

The picture perfect painted above is not easy to achieve. Not everything that is existing and working can be changed overnight. Surely, it cannot be changed until the newer versions have proved themselves. Our understanding of power optimization is yet to mature enough and I expect it to keep many people, including me, busy for a long time to come. The IEEE's Energy Efficient Ethernet Initiative is a step in the right direction. We require reconsidering many other protocols, not just Ethernet, to archive power efficiency in the network. Power optimizations in microprocessors and in wireless devices are being sought at considerable pace and we have achieved a fair level of maturity in it. Modifications to newly deployed devices are easier to accomplish than those to legacy systems. This is also a reason why it is difficult to retrofit the Internet protocols with power efficient mechanisms.

The world as a whole is understanding the benefits of going green. As a consequence, a large number of companies are adopting energy efficient practices and have started producing equipment with energy efficiency as a prime objective. IBM's Project Big Green [17], Dell's environmental stewardship program [18], Cisco's [19], Avaya's [20] and D-Link's [21] efforts along with Intel's [14] SpeedStep and AMD's [15] PowerNow technologies are all steps in the right direction. Though no one is changing the Internet's protocols as of now, we may soon see the IEEE take appropriate actions; alongside the already established Energy Efficient Ethernet Initiative.

6 Conclusion

In this seminar, we have walked through numerous avenues of power optimization pertaining to networking. This list is by no means complete, but provides enough pointers in the right direction. Power savings in each of these avenues taken individually are minimal. However, if we consider all these avenues, together with their vast deployment, the impetus of employing energy savings grows very strong. Utilization of energy has fallouts greater than just the money we have to pay for it. Energy is a shared resource and we must make every attempt to use it optimally. The burden IT infrastructure puts on the environment can be kept in check only if all of us join hands.

References

[1] Maruti Gupta and Suresh Singh. Greening of the internet. *SIGCOMM'03*, 2003.

[2] Jonathan G. Koomey. Estimating total power consumption by servers in the U.S. and the world. 2007.

[3] Jiraporn Pongsiri, Mital Parikh, Miroslova Raspopovic, and Kavitha Chandra. Visualization of internet traffic features.

[4] Bokyung Wang and Suresh Singh. Computational energy cost of TCP. *ACM SIGMETRICS (poster)*, 2003.

[5] Piyush Shivam, Pete Wyckoff, and Dhabaleswar Panda. EMP: Zero-copy OS-bypass NIC-driven gigabit ethernet message passing. 2001.

[6] FreeBSD Kernel Developer's Manual. Zero_copy(9). http://threads.seas.gwu.edu/cgi-bin/man2web?program=zero_copy&section=9.

[7] Ronny Krashinsky and Hari Balakrishnan.

[8] Nilesh Mishra, Kameswari Chebrolu, Baskaran Raman, and Abhinav Pathak. Wake-on-WLAN. *WWW2006*, 2006.

[9] Eugene Shih, Paramvir Bahl, and Michael J. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices.

[10] Francisco Blanquicet and Ken Christensen. An initial performance evaluation of rapid PHY selection (RPS) for energy efficient ethernet. *LCN*, 2007.

[11] Chamara Gunaratne and Ken Christensen. Ethernet adaptive link rate: System design and performance evaluation. *Proceedings of the IEEE Conference on Local Computer Networks.*

[12] P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, and R. Rajamony. The case for power management in web servers, 2002.

[13] Advanced Micro Devices. ACP – The Truth About Power Consumption Starts Here. http://www.amd.com/us-en/assets/content_type/DownloadableAssets/43761C_ACP_WP.pdf.

[14] Intel. Intel Core Duo processor and Intel Core Solo processor on 65 nm process datasheet, January 2007.

[15] Advanced Mirco Devices. Maximize performace-per-watt. http://enterprise.amd.com/us-en/AMD-Business/Technology-Home/Power-Management.aspx.

[16] Wikipedia. Power management. http://en.wikipedia.org/wiki/List_of_CPU_power_saving_technologies.

[17] IBM. Big Green Innovations. http://www-03.ibm.com/press/us/en/presskit/21440.wss.

[18] Dell. Dell and the Environment. http://www.dell.com/content/topics/global.aspx/corp/environment/en/index?c=us&l=en&s=gen&~ck=anavml.

[19] Cisco. Cisco Green. http://newsroom.cisco.com/dlls/2008/hd_030308.html.

[20] Avaya. Go Green at Gigabit Speeds. http://www.avaya.com/gcm/master-usa/en-us/corporate/alliances/alliance/companies/extremev30.htm&Filter=Name:Go%20Green%20at%20Gigabit%20Speeds.

[21] PRLog.Org. D-link introduces business-class green ethernet switches. http://www.prlog.org/10053273-link-introduces-business-class-green-ethernet-switches.html.

[22] Himanshu Anand, Casey Reardon, Rajagopal Subramaniyan, and Alan D. George. Ethernet adaptive link rate (ALR): Analysis of a MAC handshake protocol. *Proceedings of the IEEE Conference on Local Computer Networks*, 2006.