ISSN: 2229-6948(ONLINE)
DOI: 10.21917/ijct.2011.0049

ENERGY AWARE NETWORK: BAYESIAN BELIEF NETWORKS BASED DECISION MANAGEMENT SYSTEM

Santosh Kumar Chaudhari[1] and Hema A Murthy[2]
Department of Computer Science and Engineering, Indian Institute of Technology Madras, India
E-mail: [1]skchaudhari@lantana.tenet.res.in, [2]hema@lantana.tenet.res.in

Abstract
A Network Management System (NMS) plays a very important role in managing an ever-evolving telecommunication network. Generally an NMS monitors & maintains the health of network elements. The growing size of the network warrants extra functionalities from the NMS. An NMS provides all kinds of information about networks which can be used for other purposes apart from monitoring & maintaining networks like improving QoS & saving energy in the network. In this paper, we add another dimension to NMS services, namely, making an NMS energy aware.
We propose a Decision Management System (DMS) framework which uses a machine learning technique called Bayesian Belief Networks (BBN), to make the NMS energy aware. The DMS is capable of analysing and making control decisions based on network traffic. We factor in the cost of rerouting and power saving per port. Simulations are performed on standard network topologies, namely, ARPANet and IndiaNet. It is found that ~2.5-6.5% power can be saved.

Keywords:
Energy Aware Network Management System (EA-NMS), Next Generation Networks (NGN), Bayesian Belief Networks (BBN), Decision Management System (DMS)

1. INTRODUCTION

Global Warming and huge power requirements of Next Generation Networks (NGNs) have made energy consumption a very important issue today. In the context of NGNs, research shows that Information and Communication Technology (ICT) alone consumes 2-10% of the total consumed energy in the world [1]. The main energy consumer in the ICT field are servers, data centers, routers, cooling, fixed and mobile telephony, local area network, and printers, etc. This power consumption however excludes consumer electronics like Cell Phones and PCs [2].

Active nodes such as routers and switches in the Internet consume significant amounts of energy. The operational cost of ICT is next to the personnel cost. Within an ICT infrastructure 13% [3] power is consumed only by its network equipment. The growing demand for new multimedia services and the resulting expansion of bandwidth and traffic increases the demand for more energy. It has thus become very crucial for all industries including the IT industry to explore green computing. Energy efficiency is required for cost saving and reducing CO_2 emission from devices. Ideally for power efficiency, devices should consume energy in proportion to their load. For example, for a data center, the network infrastructure alone is responsible for 23% of overall power consumption, even disregarding the energy used for cooling the equipments [4].

A Network Management System (NMS) is used to monitor the network to maintain the performance with a focus on guaranteeing sustained QoS for applications and services. But in current systems, monitoring energy consumption by network elements particularly during the off peak periods is not addressed. This cost can be saved. Communication providers require a system that will help in providing reliable service, while at the same time cutting down the operational cost with minimal or no human interaction.

An Integrated and Intelligent Energy Aware NMS can play a vital role in saving power consumed by under-utilized elements by regularly monitoring diurnal traffic on the network. A key assumption in this paper is that all the network elements are managed elements. Hence it is possible to determine the under-utilized ports/nodes of a network. These under-utilized elements are less efficient from energy point of view even if they are primarily needed for forwarding traffic. It is possible that the volume of traffic on the port/link is so small that it can be put to sleep and thus power can be saved during the sleeping period of that port. The traffic on a sleeping port/link can be regulated and rerouted by making dynamic changes to the route using the Energy Aware NMS.

The main challenge is to figure out which port/link should go to sleep? How frequent should the port/link go into sleep/active mode? How does this affect QoS? Does the power saving and its possible consequences on other network performance parameters represent a good tradeoff? Is the NMS capable of performing this operation? If the answer is yes, then what will be the cost of extra processing power, required by the NMS server? What is the cost of rerouting traffic in terms of power?

The remainder of this paper is organized as follows. Section 2 provides a literature survey and background study on the related work. Section 3 discusses the application of BBN in the proposed energy aware NMS framework. Section 4 discusses the proposed algorithm and deployment in the network. Section 5 discusses the simulation methodology and results. Section 6 provides the conclusion of our work and future work.

2. BACKGROUND AND RELATED

The NMS primarily monitors the health of network devices and assists service providers in providing QoS guarantees. It does this by collecting different types of information like throughput, status of ports, etc., from network devices. This information helps the operator manage and improve network infrastructure. There are several standards like SNMP [5], CMIP and FCAPS defined for NMS. In this paper, we suggest that the NMS also factor in the issue of energy while evaluating the performance of a node.

Earliest work in this domain is by Gupta et al. [6]. They suggest that under-utilized network elements (either the complete router or some of its ports) can be put to sleep. They address the challenges of *sleep mode decision-making* and *sleep time prediction*. Christensen [7] stresses the need for efficient

power management in computer networks in order to save power. Gunaratne et al. [8] proposed techniques like proxying, split TCP connections and link speed scaling to conserve energy in desktop PCs and LAN switches. Gupta et al. [9] showed that the percentage of total time a link can be shutdown/put to sleep can be anywhere between 60-80%. A mechanism called Dynamic Ethernet Link Shutdown (DELS) was implemented to demonstrate this. Chiaraviglio [10] proposed some heuristic algorithms to implement the power reduction in networks by switching off network elements and links and showed that it is possible to save 10-25% energy. Gunaratne et al. [11] showed that an Ethernet link with Adaptive Link Rate (ALR) can operate at a lower data rate for over 80% of the time, yielding significant energy savings with a small increase in packet delay. A more recent work by [4] analyses traffic data from real enterprise and data center networks to simulate and test their energy saving approaches and shows that they could save energy to the order of 16%. Fisher [12] proposes creation of sleep bundles of multiple physical cables and line cards. Since identifying the optimal set of cables to shutdown is an NP-complete problem, they proposed several heuristics based on linear optimization techniques.

In our earlier work [13] we proposed a single router based framework for saving energy. A BBN was used to predict the port of a router for sleep/wake-up operation. But if a router is put to sleep this can lead to congestion in other nodes, thus violating QoS guarantees.

In this paper, we propose a centralized framework for energy awareness. Crucial to this proposal is a centralized NMS that contains information about all the nodes in the network. In the proposed solution BBN is applied on the entire network. This framework offers acceptable QoS performance with significant power savings.

3. APPLICATION OF BBN IN ENERGY AWARE NMS FRAMEWORK

In general networks are highly dynamic in nature with respect to topology and services. Networks like IndiaNet/ARPANet need to be monitored continuously for a number of performance variables like inbound traffic, outbound traffic, various delays, power consumption etc. An NMS keeps track of histories of performance variables on a regular basis. This history can be used to build statistical models of the network. In particular, a machine learning framework can enable traffic prediction.

The Energy Aware NMS framework is similar to the framework proposed [13], but is different in terms of NMS design and functionality. The NMS design is based on a centralized DMS as shown in Fig.1. The framework consists of three modules, namely, Network Management System (NMS), Decision Management System (DMS) and Configuration Management System (CMS). NMS is the central entity which interacts with DMS and CMS. All the three modules work in tandem. The NMS is based on an SNMP protocol which collects network management data using the SNMP Management Information Base (MIB) of network elements. To monitor power consumption, traffic load information and QoS related data is required. The NMS collects this data from the network and sends it to DMS. The DMS analyses the data and takes the decision, which is then sent to CMS to make final changes in the network elements.

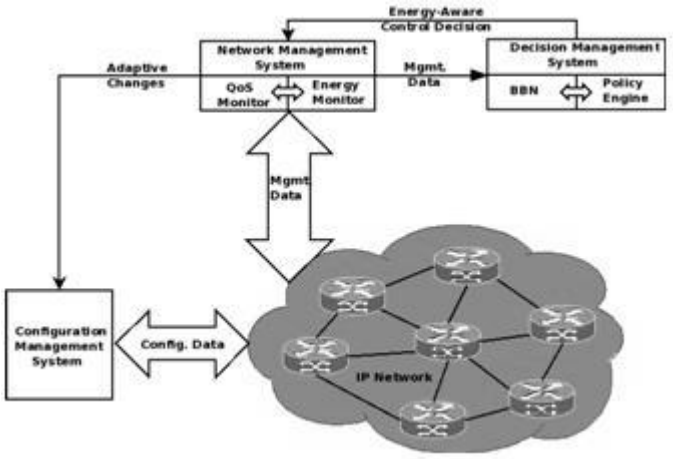

Fig.1. The Energy Aware NMS Framework [13]

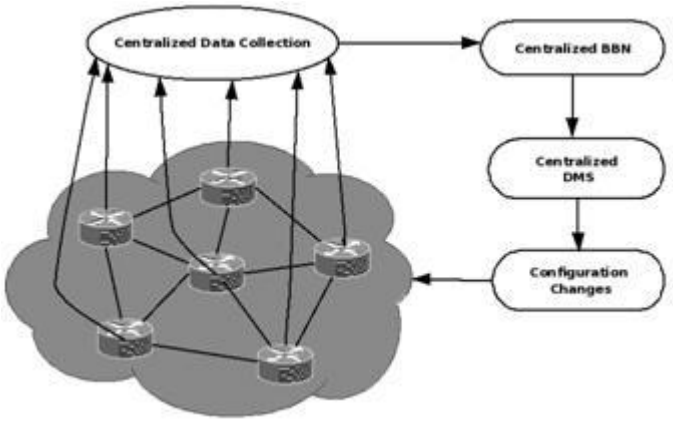

Fig.2. DMS with Centralized BBN

The energy aware NMS predicts the future consumption of energy by network elements and thereby controls the mode of operation of a port, namely sleep and wakeup modes.

Machine learning techniques are capable of modeling the system behaviour through a learning process. The system behaviour is learnt based on observations made on a network over a period of time. Once the model is trained, it is capable of predicting the future system behaviour.

Since the volume of network data gathered by NMS is large, and the proposed approach requires automated systems which can learn efficiently, a machine learning technique called Bayesian Belief Networks (BBN) [14]-[16] is used. BBN is swiftly able to learn network traffic behaviour and predict the future network behaviour accurately. The prediction is based on the aggregate traffic load. This prediction forms the basis for pro-actively putting a particular network element or its ports into sleep/wake-up mode. In this framework we rely on the existing routing protocol namely RIP/OSPF.

The assumption we have made in this framework is that, if a particular port/node's running states gets changed, the rerouting will be handled by the routing protocol itself. The QoS as well as other explicit hard-coded requirements of network (e.g.

certain ports should not be put into sleep) can be maintained by defining a policy. Thus, while the BBN provides the ability for the DMS to intelligently take decisions based on the policy, the policy engine validates these decisions based on QoS and other hard-coded requirements of the network.

Centralized DMS: In Centralized DMS Fig.2, a single DMS runs in tandem with an NMS server and operates on the entire network. The data collected by an NMS will be used by the DMS and the decision taken by the DMS will affect the entire network globally. The main challenges in this model are the delays encountered in decision-making and the final configuration changes in network while maintaining QoS.

The DMS requires a sufficient amount of training data, computation and processing power. So, the DMS could instead be deployed on a separate server along with the NMS server. In the Centralized DMS, all the data and resources are available at the server itself and the decision made by the DMS will have a global effect. A centralized DMS is the key contribution in this paper. The algorithm for the Centralized DMS is discussed in the following section.

4. ALGORITHMS AND DEPLOYMENT

The implementation of the DMS is a little complex and is based on some assumptions. We have assumed that the network topology corresponds to the connectivity of nodes in the network. The DMS is fully aware of network topology and policies. The main part of DMS is the BBN module and it requires a significant amount of data to learn the network behaviour. The training of BBN model is done offline. Once the model trained, it is deployed in the network. Decisions will be made by the BBN module and it needs to be re-trained over a period of time.

The other main part of DMS is the policy engine. The DMS needs to define different policies to handle different type of QoS provisioning and performance requirements. The policy can for example be: if any link is very critical and it can cause network disconnection or violation of Service Level Agreement (SLA), it cannot be put into sleep mode. Putting some ports to ***sleep/off*** mode constitutes a violation of the guaranteed QoS SLA. Even in under-utilised conditions a particular port cannot be put to sleep mode for long durations. Further, as indicated in [22], the QoS guarantees have to be provided end-to-end: addressing the power in a single router may save power at the router but can lead to congestion on the other routers.

The CMS works along with the DMS. It needs instructions from the DMS in the form of {**NODE: PORT: SLEEP /WAKEUP**}. Once it receives the instruction, it makes the final changes in the physical network.

As discussed, there are so many issues which we have tried to capture in this proposed framework. The working algorithm of the centralized DMS is as presented in Algorithm 1.

Algorithm 1 Centralized DMS

Step1: **repeat**

Step2: **repeat**

Step3: **For** *each node n=1, 2… n in the network*

Step4: **repeat**

Step5: **For** *each port i=1,2,....p, determine the average link utilization.*

Step6: $U_i = (total_i * 100)/ C_i$

Step7: *(Where* $total_i = in_i + out_i$*)*

Step8: **Until** *No port left*

Step9: *The dependency graph for the BBN is obtained using this data and the physical connectivity is obtained from the routing table.*

Step10: *Find the under-utilised or over-utilised {crossing min threshold or max threshold value} ports from a set of ports.*

Step11: *Policy engine verifies whether the selected port is violating any policy {e.g., Port is not connecting a critical link and can cause network disconnection}.*

Step12: *If there is no violation of policy, apply the decision to the network by CMS.*

Step13: *Keep updating the BBN model at regular intervals of intervals {in our case its 100 sec}.*

Step14: *Create discrete data in form of \{low, med, high\} for each port to train/re-train BBN.*

Step15: *No node left*

Step16: *Update the BBN model at regular intervals.*

Step17: **Until** *Simulation Over*

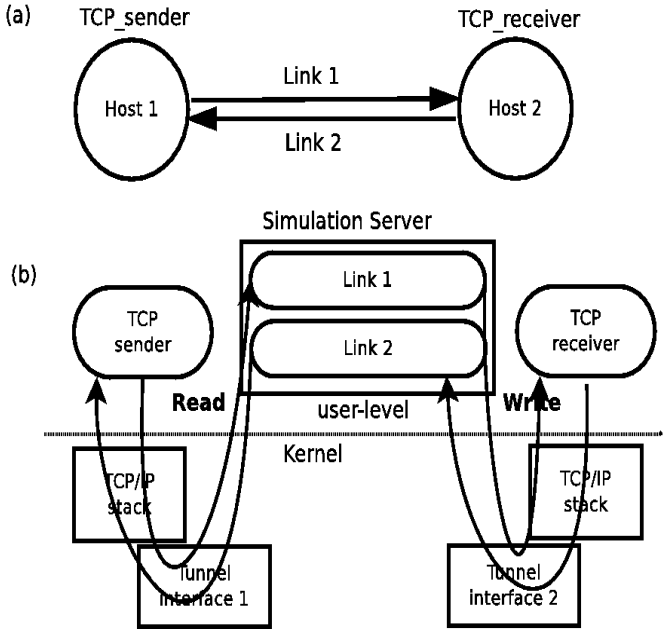

Fig.3. (a) A Simple TCP/ IP Network and (b) Implementation in NCTUns

Table.1. Network Data Which Form The BBN Nodes

Node name	States	Details
Utilisation_x_y (x = node no., y = port no.)	low med high	low (0-40%) med (41-69%) high (70-100%)

5. SIMULATION, RESULTS & DISCUSSION

5.1 NETWORK SIMULATOR & BBN TOOLS

The simulation has been performed using the National Chiao Tung University network simulator (NCTUns) [17]. NCTUns is a network simulator and emulator executable on Linux with fully-integrated GUI environment. Using this simulator, a user can create and edit a network topology, configure the protocol modules used inside a network node, specify paths for mobile nodes, plot performance curves, play back animations of logged packet transfers, etc. It accepts any UNIX application program as a traffic generator program without any modification. It also enables the use of any real-world UNIX network configuration and monitoring tools, such as an NMS. It can simulate various networks and network devices.

One of the key technologies that makes NCTUns unique is kernel re-entering simulation methodology using tunnel network interface. A tunnel network interface, available on most UNIX machines, is a pseudo network interface that does not have a physical network attached to it. The functions of a tunnel network interface, from the kernel's point of view, are no different from those of an Ethernet network interface. A network application program can send or receive packets through a tunnel network interface, just as if these packets were sent to or received from a normal Ethernet interface. Fig.3(a) shows a simple TCP/ IP network and Fig.3(b) its implementation in NCTUns [18].

To implement the BBN module of DMS, we used Hugin Researcher [19]. The learning and inference algorithms used in our framework were implemented using JAVA APIs.

5.2 SIMULATION SETUP

For simulation, we have chosen two topologies, IndiaNet (Fig.4) and ARPANet (Fig.5) to study the performance of DMS. The IndiaNet and ARPANet have 20 and 21 router nodes respectively. Both networks are fully connected in topology. The links between nodes are bidirectional in nature and have 50 Mbps capacity. There are a number of routers and sources/sinks to send/receive traffic.

The number of sources and sinks for IndiaNet is (32, 13) and for that of ARPANet is (27, 27) respectively. The network data collected by the NMS is made available to the DMS.

5.3 TRAFFIC CHARACTERISTICS

To provide maximum randomness while simulating network traffic, the traffic sources were modeled as Poisson processes. The different parameters used for generating Poisson traffic are given in Table.2. In our earlier work [13], we created unidirectional traffic between some nodes, but in this work we have chosen bidirectional traffic. For simulating a large network, we have selected some nodes that have high traffic, for example all the metro cities in India. We have selected these cities based on the assumption that these metro cities have more number of internet users and act as gateways for other nodes. For e.g., node 3, 4, 11, 16 and 20 are heavily loaded compared to other nodes. However, for ARPANet, the source and sink have been randomly selected because we don't have enough information about the network. So, there is a difference in traffic flow between the two types of networks.

Table.2. Characteristics of Traffic Sources

Parameter	Value
Packet Inter-arrival Time(s)	Exponential (0.001)
Packet Size (bytes)	Exponential (1200)
Traffic Start Time (s)	Constant(0.1)
Traffic ON state time(s)	Constant(100)

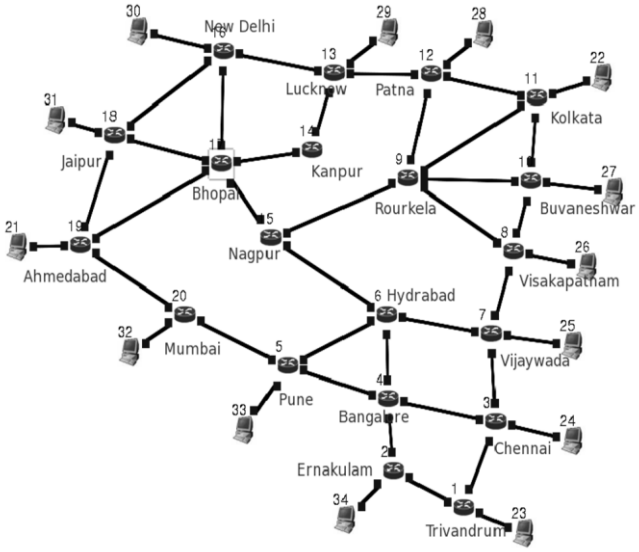

Fig.4. India Network (IndiaNet) Topology

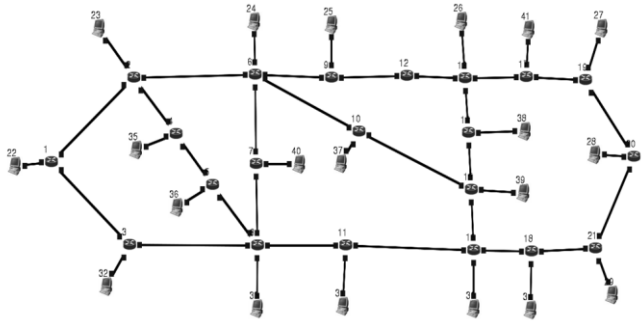

Fig.5. ARPA Network (ARPANet) Topology

5.4 EXPERIMENTATION DETAILS

We have chosen the simulation time for the experiment in between 2200-2500 sec. The first 1000 sec of simulation data is used to train the BBN. Then, at regular intervals of 100 sec, data is provided to the BBN to update the model. The BBN learns the structure from the training data using PC algorithm [23], [24] and then builds the model using Expectation Maximization (EM) algorithm [25]. The network device has many MIB variables whose details can be very useful to the network administrator with respect to network management. But for this work we focus only on a few MIB variables, namely: In-bound traffic, Out-bound traffic, In-Out-traffic.

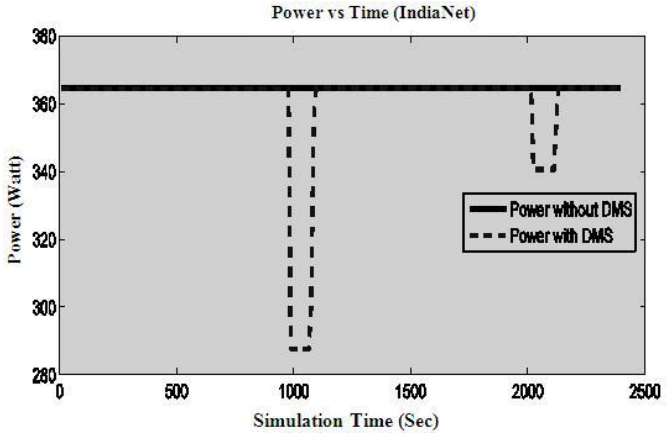

Fig.6. Power vs Time (IndiaNet)

Table.3 and Table.4 shows the results for IndiaNet and ARPANet respectively. The approximate power saving is calculated based on the power rating of an enterprise switch [4] which is P_{active}=0.48 watt and P_{sleep}=0 watt for a 100 Mbps link. The power rating is assumed to be the same for both the 100 Mbps and 50 Mbps ports. Hereafter, all power calculations will be based on this power rating. The total number of ports is 76 and 69, for IndiaNet and ARPANet, respectively.

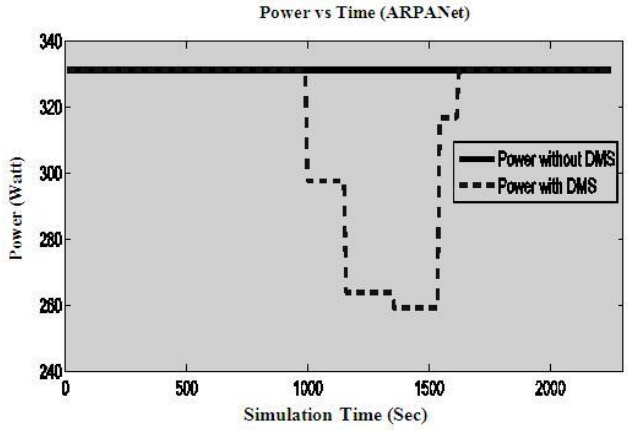

Fig.7. Power vs Time (ARPANet)

From the experiment on IndiaNet, we found that the DMS made decisions to make 32.01% of the ports to sleep on an average (Table.3). The results presented in Table.3 and Table.4, are ensemble averaged over 4-5 experiments. The total simulation time for IndiaNet was 2450. The average number of sleeping ports was found to be 24.33. The average sleeping time per port was found to be 196.87 sec.

The power calculation done based on duration of simulation and power consumption of a port. So,

*Total Required Power = Total Simulation Time * Total No. of Ports * 0.48 (watt)*

*Total Power Saving = Total Sleeping Time * 0.48 (watt)*

The power saving per port was found to be 94.5 W. Since traffic has been logged for an interval of 10 sec, the power was also calculated for the same interval. So, in the case of no DMS, the total power consumed by IndiaNet was found to be 89376 W. After using DMS, total saving of power was found to be 2299.2 W. The Fig.6 shows the total power consumption without DMS vs power consumption with DMS for a IndiaNet. From the graph, most of the ports seem to be sleeping during the 1000-1300 sec and 2000-2200 sec duration.

Table.3. Power Saving by IndiaNet[1]

ST (sec)	TNP	TNSP	TST (sec)	AST /port (sec)	APS /port (W)	PSM (mJ)
2450	76	24.33	4790	196.87	94.5	1.95

Table.3. Power Saving by ARPANet[1]

ST (sec)	TNP	TNSP	TST (sec)	AST /port (sec)	APS /port (W)	PSM (mJ)
2270	69	23	10060	437.39	209.95	1.85

After repeating the same experiment on ARPANet, we found that DMS made 33.33% ports to sleep on an average (Table.4). The average number of sleeping ports is 23. The average sleeping time per port is 437.39 sec. The average power saving per port in this case is 209.95 W. Again traffic was logged at the same interval as IndiaNet. So total power consumed by ARPANet without DMS was found to be 75182.4 W and after using DMS, total power saving was found to be 4828.8 W. Fig.7 shows the total power consumption without DMS vs power consumption with DMS for a ARPANet. For ARPANet, most of the ports are sleeping during the 1000-1700 sec duration.

It may be noted that ARPANet is having significantly more power saving than IndiaNet. The reason for this is that in the case of IndiaNet some of the nodes carry more traffic than the others, for example, the nodes in the four metros in India. Also the number of sources and sinks in ARPANet is equal whereas this is not the case with IndiaNet. From Fig.6 and Fig.7, it is clear that the power consumption decreases when some ports are sleeping and increases when the ports are active.

The average running time of DMS was 1838.38 ms. The cost of updating the routing table was not taken into consideration because this is part of the routing protocol itself. An SNMP packet of size 564 bytes is used to send messages to a node/router to make a link sleep/wake-up. Therefore, it adds very little cost to the existing traffic when compared to the actual network traffic.

The rerouting cost is calculated based on energy per bit in network devices, which is 10nJ per bit for core routers [26]. The average rerouting cost per port for IndiaNet is 0.95 W and for ARPANet is 2.64 W. The difference in power saving and rerouting cost is due to the different power ratings we have chosen for the networks. If we scale the rerouting cost with respect to the power rating used to calculate the power saving, IndiaNet saves more power than ARPANet. The power consumed by a SNMP packet is 48.68 µJ. Power consumed by SNMP messages can be neglected as they are negligible.

[1]Abbreviation-ST: Simulation Time, TNP: Total no. of ports, TNSP: Total no. of sleeping ports, TST: Total sleeping time, AST: Average sleeping time, APS: Average power savings, PSM: Power by SNMP messages.

6. CONCLUSIONS

Although the result presented here is a simulation based on two well known topologies, the algorithm can work on all kinds of networks. The DMS presented here is capable of prediction in real time. From the result we have been able to demonstrate that by using a BBN based centralized DMS, 20-33% ports can be put to sleep without affecting the network performance and services and save 2.5-6.5% power. As future work, we plan to implement DMS on a test bed for actual performance measurement in different network conditions. Further, the effect of rerouting on QoS needs to be studied.

ACKNOWLEDGEMENTS

This work has been supported by the EPSRC funded India-UK Advance Technology Centre in Next Generation Networking.

REFERENCES

[1] "An inefficient truth", Global Action Plan Report, http://www.globalactionplan.org.uk/, 2007.

[2] http://www.gartner.com/it/page.jsp?id=503867

[3] Source: "Energy consumption by office and telecommunications equipment in commercial buildings", Vol.1, 2002.

[4] P. Mahadevan, "Energy Aware Network Operations", *in Proceedings of IEEE INFOCOM*, pp 1-6, 2009.

[5] D. Harrington, R. Presuhn, B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", *RFC 3411, IETF*, 2002.

[6] M. Gupta, S. Singh, "Greening of the Internet", *in Proceedings of ACM SIGCOMM*, pp 19-26, 2003.

[7] Christensen K, Nordman B, Brown R, "Power management in networked devices", *IEEE Computer*, Vol.37, No.5, pp.91–93, 2004.

[8] C. Gunaratne, K. Christensen, B. Nordman, "Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying,Split TCP Connections, and Scaling of Link Speed", *Intl. Journal of Network Management*, Vol. 15, No. 5, pp. 297-310, 2005.

[9] M. Gupta, S. Singh, "Dynamic Ethernet Link Shutdown for Energy Conservation on Ethernet Links", *in Proceedings of IEEE ICC*, pp 6156-6161, 2007.

[10] L. Chiaraviglio, "Energy-Aware Networks: Reducing Power Consumption by Switching Off Network Elements", GTTI 2008. Available at:
www.gtti.it/GTTI08/papers/chiaraviglio.pdf.

[11] C. Gunaratne, K. Christensen, B. Nordman, S. Suen, "Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR)", *IEEE Transactions on Computers*, Vol. 57, No.4, pp. 448–461, 2008.

[12] W. Fisher, M. Suchara and J. Rexford, "Greening Backbone Networks: Reducing Energy Consumption by Shutting Off Cables in Bundled Links", *ACM SIGCOMM Workshop on Green Networking*, pp. 29 – 34, 2010.

[13] A. Bashar, G.P. Parr, S.I. McClean, B.W. Scotney, M. Subramanian, S.K. Chaudhari, T.A. Gonsalves, "Employing Bayesian Belief Networks for energy efficient Network Management", *National Conference on Communications (NCC)*, Vol., No., pp.1-5, 29-31, 2010.

[14] D. Heckerman, "*A Tutorial on learning with Bayesian networks*", Learning in Graphical Models, M. ~Jordan, ed. MIT Press, Cambridge, MA., 1999

[15] K. B. Korb and A. E. Nicholson, "*Bayesian Artificial Intelligence*", 1st Ed., CRC Press Co., UK., 2003.

[16] A. Bashar, G.P. Parr, S. I. McClean, B.W. Scotney and D. Nauck, "BARD: A novel application of Bayesian reasoning for proactive network management", *in Proceedings of the 10th Annual Postgraduate Conference on Telecommunications, Networking and Broadcasting (PGNET 2009)*, Liverpool, UK.

[17] S.Y. Wang, C.L. Chou, C.C. Lin, "The Design and Implementation of the NCTUns Network Simulation Engine", *Elsevier Simulation Modelling Practice and Theory*, Vol.15, No.1, pp. 57 – 81, 2007.

[18] S.Y. Wang and H.T. Kung, "A Simple Methodology for constructing Extensible and High-Fidelity TCP/IP Network imulators", *IEEE INFOCOM'99*, New York, Vol.3, pp.1134-1143.

[19] ``www.hugin.com". Accessed March 2011.

[20] "http://www.tenet.res.in/graph/indianet.html". Accessed March 2011.

[21] http://www.optical-network.com/topology.php. Accessed March 2011.

[22] J.H. Saltzer, D.P. Reed and D.D. Clark, 1984, "End-to-end arguments in system design", ACM Trans. Comput. Syst.2, 4, 277-288.%
DOI=http://doi.acm.org//10.1145/357401.357402.

[23] P. Spirtes, C. Glymour, R. Scheines, "Causation, Prediction and Search, MIT Press", Second edition, 2001.

[24] H.~Steck, "Constrained-based structural learning in Bayesian networks using finite data sets", Phd Thesis, Institut fűr der Informatik der Technischen Universität München, 2001.

[25] F. Jensen, "*Bayesian Networks and Decision Graphs*", 2nd Ed., Springer Co., NY, USA, 2007.

[26] R. Tucker, "A Green Internet", (Plenary Paper), *IEEE Lasers and Electro-Optics Society*, (LEOS), 21th Annual Meeting, Newport Beach, USA, 2008.