Distributed and Load Adaptive Energy Management Algorithm for Ethernet Green Routers

Mohamad Khattar Awad[1], Phone Lin[2], Gi-Ren Liu[3]

[1] Department of Computer Engineering, Kuwait University, Kuwait
[2] Department of Computer Science and Information Engineering, National Taiwan University, Taiwan
[3] Department of Mathematics, National Cheng Kung University, Taiwan
mohamad@ieee.org, plin@csie.ntu.edu.tw, girenliu@gmail.com

Abstract

The existing Ethernet networks are designed with high redundancy and over-dimensioning so they can provide reliable services during peak traffic demand periods. However, this has increased the total energy consumption and operational cost. In this paper, we propose an energy saving algorithm (ESA) to reduce the energy consumption of green routers by considering the buffer status and the traffic load. We develop a Network Simulator, version 2, (NS-2)–based simulation model for ESA to evaluate its performance with respect to real traffic traces. Performance bounds of the proposed algorithm are derived. Numerical evaluations are conducted to verify the accuracy of the simulation model against derived bounds. Performance evaluations demonstrate that the proposed algorithm outperforms candidate algorithms, thereby providing greater energy savings with an acceptable packet delay and loss. We show that the introduced delay is bounded by an upper bound that is slightly larger than half of the sleep timer. Furthermore, performance comparisons are extensive and detailed, thus providing insights into the performance of different energy saving functions considered by the candidate algorithms.

Keywords: Green router, Energy-efficient networks, Energy management algorithms, Network simulator (NS-2)

1 Introduction

Interest in energy efficient networking has both ecological and economic motivations. The accumulation of greenhouse gases has reached an alarming level according to many climate science reports. Information and communication technology (ICT) contributed approximately 2% of the total CO_2 emission from human activities [1-2]. This corresponds to 830 Mega-ton-CO_2 and is expected to grow to 1.43 Giga-ton-CO_2 in 2020, which is approximately 2.7% of the total CO_2 footprint. Along with the rapid growth of Internet usage and demand for higher data rates in the Internet of Things (IoT) or Machine-to-Machine (M2M) networks, not only environmental but also the financial costs are expected to grow. Taking the European Telecommunications network infrastructure as an example, the energy cost for such networks was approximately $2 billion U.S. in 2010 and is expected to grow to $4.3 billion U.S. in 2020 [3].

Routers constitute major part of today's Internet infrastructure. Therefore, re-engineering routers to make them green, i.e., energy efficient, would have a significant impact on the energy efficiency of the Internet. An analytical model to evaluate the impact of various energy saving approaches on network performance supports this claim and shows that a large energy saving margin can be exploited at the data plane [4-7]. Along the same line, [8] shows that a network energy saving of 12% can be achieved only by changing the migration sequence of green routers. Savings as high as 70% of the router energy consumption is brought by bypassing the table look-up operations under a centralized approach proposed in [9] and [10]. There are opportunities for significant energy savings at both the control and data plane, let alone the operational plane, i.e., power supply, cooling, etc. This work contributes to the optimization of the energy consumption of green routers at the data plane.

The previous works that focused on developing distributed algorithms [11-18] are summarized as follows. In [11], Gupta and Singh proposed a distributed approach to selectively put network interfaces to sleep during low demand periods. In other words, an interface is put to sleep if the inter-arrival time is longer than a pre-defined threshold. Packets arriving during the sleep period wake a sleeping interface. This wake-up mechanism was also adopted in [12] and IEEE 802.3az standard [19], which is also known as Energy Efficient Ethernet (EEE). In [13], authors predicted that the adoption of EEE may result

*Corresponding Author: Mohamad Khattar Awad; E-mail: mohamad@ieee.org
DOI: 10.3966/160792642018051903014

in $400 million saving per year in the U.S. alone. An analysis of the application of EEE to Real Time Ethernet (RTE), which is a common factory communication system, shows that a considerable power saving can be achieved in RTE despite its tight timing requirements [20, 21]. In [22], Chatzipapas et. al. analytically modeled the behavior of EEE links with coalescing using M/G/1 queues with sleep and wakeup periods. In [23], Rodríguez-Pérez et. al. extended EEE to address the issue of energy efficiency of link aggregates that are commonly used to raise the capacity of network connections.

Based on the energy saving approaches and performance findings presented in [15], an opportunistic energy saving algorithm with different conditions for switching the interfaces *on* or *off* is proposed in [17]. The interface is switched *off* when the buffer is empty to avoid unnecessary delay in transmitting queued packets. In addition, the interface is switched *off* when the predicted sleep duration is longer than a predefined threshold. The threshold is designed such that the interface sleeps for sufficiently long period to guarantee an energy saving. To avoid high *on-off* oscillation, interfaces are reactivated when the number of packets that arrive during a sleep period exceeds a given threshold. The implementations reported in [17] queue arriving packets until the activation condition is satisfied; this shifts the latency from packets that arrive before the sleep period to those that arrive after it. Unfortunately, packets arriving during the sleep period remain subject to loss due to buffer overrun.

The latency introduced by queuing packets during sleep periods can be eliminated by assigning shadow ports or shadow devices to serve the packets arriving at the sleeping ports [16]. Three energy saving algorithms were proposed in [16]; two of them assign a limited number of ports to clusters of sleeping ports, and the third assigns a low power device to handle the ingress packet traffic arriving at either a cluster of ports or a high-power device. The three algorithms buffer egress packets during sleep periods. The first algorithm predicts the number of packets arriving over the next time window and puts the interface to sleep if the predicted number of packets is less than a predefined threshold. The second algorithm switches the port *off* at pre-scheduled sleep time irrespective of the traffic pattern. The third algorithm powers down the high power device and relays the traffic to a low-power shadow device. Whereas the first algorithm adapts to the traffic load, the second one is more aggressive and introduces latency and packet loss. Technical and monetary overhead makes the third algorithm impractical.

The recently developed approaches for reducing energy consumption in networks have considerable potential. Nevertheless, these attempts have been restricted to networks of one type of routers and suffer from long packet delay, considerable packet loss, frequent *on-off* oscillation or high financial cost. The primary focus of this paper is to optimize the energy efficiency of Ethernet infrastructures that consist of both traditional and energy efficient, i.e., green, routers as illustrated in Figure 1. An energy saving algorithm (ESA) to reduce the energy consumption of such networks is proposed. The algorithm runs on green routers and takes advantage of the availability of traditional routers to balance between packet delay and loss, as well as switching oscillation while reducing energy consumption. Furthermore, it is distributed and backward compatible and does not incur any additional economic cost. The sub-components of our green router interface alternate between three states: *active* state, *sleep* state, and *rescue* state. In the *sleep* state, packets are buffered for future transmissions. To prevent packets from being dropped due to buffer overflow, the *rescue* state is activated as the buffer occupancy grows beyond a predefined threshold. In the *rescue* state, arriving packets are rerouted to other active traditional routers. The contribution of this research are summarized as follows:

- First, the proposition of a low loss mode of operation to protect packets against loss in infrastructures consisting of traditional and green routers.
- Second, the definition of a distributed three-state buffer management algorithm for green routers.
- Third, the development of an analytically verified NS-2–based simulation model for buffer management algorithms of green routers.

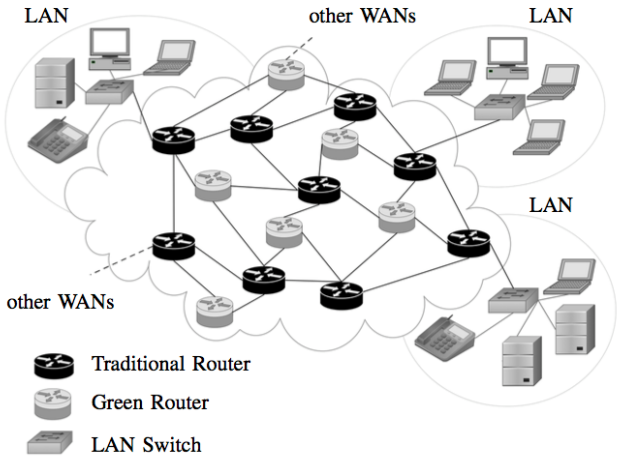


Figure 1. An illustration of a network consisting of both traditional and green routers

The remainder of this paper is organized as follows. Section 2 introduces the network model consisting of both energy efficient and traditional routers. In addition, a generic router architecture of traditional routers is also presented in Section 2. The proposed algorithm is presented in Section 3. In Section 4, we present analytical models to verify the simulation

model described in Appendix A, and provide insights into the ESA properties. The performance of the proposed algorithm is evaluated in Section 5 and is followed by conclusions in Section 6. In Appendix A we present the NS-2 based simulation model and validate it against derived bounds in Appendix B. The source code of the NS-2 simulation experiments is available at www.mohamadawad.com.

2 Network Model

We consider an Ethernet network that consists of traditional routers and energy efficient routers, i.e., green routers, as illustrated in Figure 1. Traditional routers and energy efficient routers are assumed to have the same basic architecture; however, the energy efficient routers manage the energy consumption of their interfaces based on the proposed ESA. Energy efficient routers are connected to routers of the same type or to traditional routers that do not support energy efficient routing. In such scenarios, on one side of a physical link, the transmitter (Tx) can be switched off while the receiver (Rx) on the other side remains on. Additionally, on one side of the physical link, Tx can be off while Rx might be on to service packets that are transmitted from an active core router. To facilitate such independent operations of Tx and Rx on one side of the physical link, a detailed router architecture is considered where Tx and Rx at the same interface operate independently.

Figure 2 shows the major components of a router, which includes a routing processor, a switch fabric, and a number of interfaces. The routing processor executes the routing protocol (e.g., OSPF [24], RIP [25], and BGP [26]), while maintaining the routing table and performing network management functions.

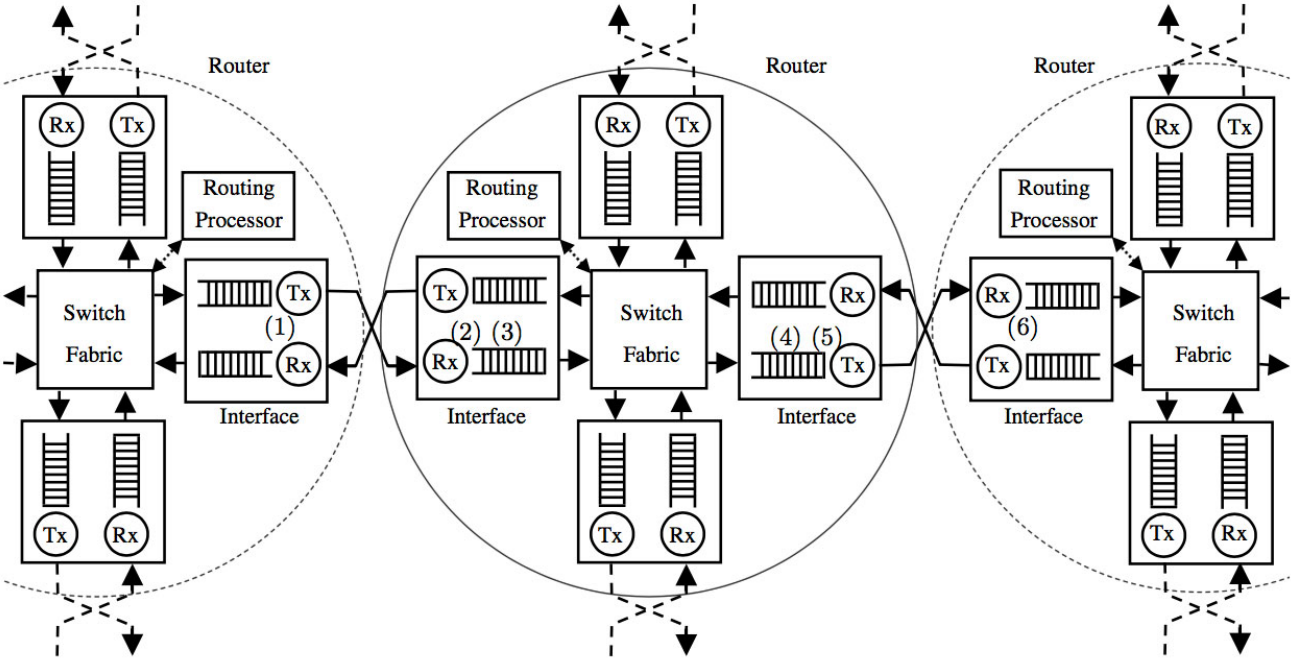


Figure 2. Generic architecture of three routers and links among them. The illustration shows each of the router subcomponents: switch fabric, routing processor and interface

Based on the routing table lookup results, the switching fabric forwards ingress packets to the next hop via an Rx (Figure 2-(2)) connecting to the appropriate Tx (Figure 2-(1)). The switch fabric can be implemented in various ways, such as shared memory, shared medium, or a crossbar [27]. An interface consists of Tx, Rx and their buffers. The interfaces of two neighboring routers are connected through a bi-directional physical link. The Rx executes physical layer functionality to terminate an incoming physical link to a router and performs data link layer functionality to communicate with the Tx on the other side of the physical link (e.g, Figure 2-(1) and (2)).

Both the core and the energy efficient routers process packets as follows. Upon receiving a packet, the Rx buffers the packet in the Rx buffer (Figure 2-(3)). The switch fabric forwards buffered packets to the Tx buffer (Figure 2-(4)) where they are queued until they are transmitted by Tx (Figure 2-(5)). The Tx then transmits packets on the outgoing link by executing the data link and physical layer protocols with the Rx on the other side of the physical link (Figure 2-(6)).

The considered router architecture is flexible because it allows connecting routers of different types, e.g., energy efficient and traditional routers. We depict a practical scenario of real networks where energy efficient routers are introduced into existing networks of traditional routers. Moreover, the architecture is generic, which makes ESA widely applicable.

3 Proposed Energy Saving Algorithm

Putting some or all of the router components into sleep mode has shown to be effective in reducing its energy consumption [11-18, 28-29]. The amount of energy conserved by switching *off* a router subcomponent is highly dependent on the router architecture and implementation. Reducing the energy consumption of these subcomponents by putting them to sleep during low utilization periods is the basis of the proposed ESA.

Note that the switch fabric cannot be turned *off* unless all ingress and egress ports are *off*. The algorithm maintains two state machines, S_{Tx} and S_{Rx}, to operate Tx and Rx, respectively. Whereas the S_{Tx} machine changes state based on buffer occupancy and timer status, the S_{Rx} machine changes state based on S_{Tx} state and timer status. A sleep timer denoted by T_s is maintained to control the time period over which Tx remains in either the *sleep* or *rescue* state. Figure 3-top shows the transitions of S_{Tx}, which consists of three states: *active* state, *sleep* state, and *rescue* state. The three states and their transition conditions are described as follows.

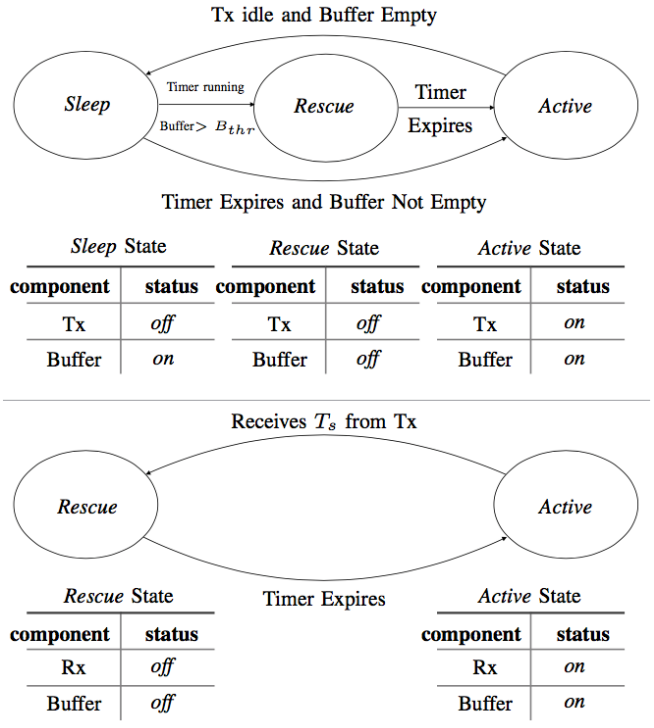


Figure 3. Representation of the state machines S_{Tx} (top) and S_{Rx} (bottom)

- In the *active* state, both the Tx and Tx buffer are *on* to process packets. S_{Tx} remains in this state until Tx is detected to be idle, which implies that the Tx buffer is empty. Under this condition, S_{Tx} changes from *active* state to *sleep* state, and Tx sends a signaling message carrying T_s to Rx on the other side of the link. The signaling message also triggers S_{Rx} to move to rescue state. The time required to switch from *active* state to *sleep* state is denoted by $T_t^{s\to a}$.
- In the *sleep* state, the Tx is switched *off* while the Tx buffer remains *on*. In this state, switching the Tx *off* does not trigger route re-calculation and the routing tables are not updated. Thus, the energy consumed by the Tx is reduced without affecting the existing routing paths. S_{Tx} remains in this state as long as the timer does not expire, and the buffer occupancy is less than a predefined threshold, B_{thr}. However, if the buffer occupancy exceeds B_{thr} while the sleep timer is still running, S_{Tx} changes its state to the *rescue* state to avoid packet loss. When the timer T_s expires, S_{Tx} changes state back to *active* if there are queued packets to process; otherwise, S_{Tx} returns to the *sleep* state. The time required for the circuitry to switch from the *sleep* state to the *active* state is denoted by $T_t^{s\to a}$.
- In the *rescue* state, both the Tx and the Tx buffer are turned *off*. Neighboring routers are informed of the link disconnection and a new routing path is established. The routing processor runs the routing protocols (e.g., OSPF [24], RIP [25], and BGP [26]) to find alternative routes. For packets that arrive at the interface during the *rescue* state, the routing processor forwards the packets to other available interfaces based on the re-established route. When the T_s timer expires, S_{Tx} changes the state from *rescue* to *active*. The time required for this transition equals to $T_t^{s\to a}$. Therefore, the Tx and the Tx buffer are turned *on*, the T_s timer is reset, and Tx starts transmitting buffered packets.

While S_{Tx} is in either the *sleep* or the *rescue* state, Tx is *off* and Rx on the other side of the link does not receive packets. Hence, both Rx and the Rx buffer can be switched *off*.

The state machine S_{Rx} on the other side of the link alternates between two states: the *active* state and the *rescue* state. Figure 3-bottom shows a representation of the S_{Rx} state machine. In the *active* state, both Rx and the Rx buffer on the other side of the link are *on* and can thus receive packets. Upon receipt of T_s from Tx, the Rx starts the timer and processes queued packets before S_{Rx} changes the state to *rescue*. In the *rescue* state, both Rx and Rx buffer remain *off* until the timer expires. When the timer expires Rx and Rx buffer are turned *on*, and S_{Rx} switches to the *active* state.

The state machines S_{Tx} and S_{Rx} on one side of the physical link operate independently from those on the other side. In other words, the state of a given interface sub-component (i.e., Tx, Rx, Tx buffer or Rx buffer) does not affect the state of other subcomponents of the same interface, which provides flexibility in switching sub-components *on* or *off*, and a higher energy saving margin can be achieved. Furthermore, this allows for connecting an energy efficient router to either other

energy efficient routers and/or traditional routers that do not support such energy saving mechanisms. In scenarios where the energy saving router is linked to a traditional router, the Tx and Tx buffer at this router can be switched *off* while the Rx and Rx buffer at the same interface remain *on* to serve packets arriving from the traditional router.

4 Analytical Models

In this section, we proposed analytical models to verify the simulation model proposed in Appendix A, and provide insights into the ESA properties, particularly, those associated with energy saving, packet delay and *on-off* oscillation overhead. A lower bound of the energy savings and an upper bound of the transmission delay introduced by ESA are derived. Furthermore, a new metric, the switching overhead, is obtained to capture the algorithm's impact on the *on-off* oscillation of managed subcomponents.

We consider a queuing system in which packets arrive in batches following a compound Poisson process with rate λ. The size of successive arriving batches is modeled by a random variable denoted by *Z*, and its first and second moments are $E[Z]$ and $E[Z^2]$, respectively. The system consists of a single server implementing an "exhaustive service policy." That is, the server does not switch to the *sleep* state unless the buffer is empty [30]. The service time *X* to transmit a packet has a general probability distribution with the first moment $E[X]=\frac{1}{\mu}$ and a second moment $E[X^2]$.

In ESA, the setup of the sleep timer has a significant impact on the performance. A longer sleep time forces the interface to sleep longer (i.e., more energy is potentially saved). Furthermore, it is more likely that S_{Tx} changes its state to *rescue*, and extra signaling is required for the reestablishment of the routing path. Conversely, shorter sleep time results in higher rate of *on-off* oscillation and thus higher energy consumption. In our analysis, we consider both deterministic and random sleep timers. Let the random sleep timer be a random variable *T* with general distribution function $F_T(\tau)$, first moment $E[T]$, and second moment $E[T^2]$. Additionally, denote the Laplace Stieltjes Transform (LST) of *T* by $T^*(s)=\int_{\tau=0}^{\infty} e^{-s\tau} dF_T(\tau)$. In the following, we derive performance metrics for both the random sleep timer *T* and the deterministic sleep timer T_s.

4.1 Lower Bound for Energy Saving Ratio

The energy saving ratio denoted by $R(B_{thr})$ captures the fraction of time over which Tx remains idle for a given buffer threshold B_{thr}. ESA exploits a portion of the energy saving margin by switching *off* the Tx in the *sleep* state. Energy savings are further increased by keeping Tx buffer *off* in the *rescue* state. The energy saving ratio lower bound denoted by $\breve{R}$ models the fraction of time Tx remains only in the *sleep* state. For an infinite buffer size and infinite B_{thr}, the transition condition to the *rescue* state is never satisfied and S_{Tx} alternates only between *active* and *sleep* states. Therefore, the lower bound of the energy saving ratio can be written as,

$$\breve{R}=\lim_{B_{thr}\to\infty} R(B_{thr}). \tag{1}$$

Let *I* be the random duration of idle periods and *A* be the random duration of *active* periods as shown in Figure 4. This alternation between *active* and *sleep* states over *A* and *I* periods generates an alternating renewal process [31]. Thus, we have

$$\frac{E[A]}{E[I]}=\frac{\rho}{1-\rho}. \tag{2}$$

where $\rho=\frac{\lambda}{\mu}$ denotes the system utilization and the probability that the system is busy. Note that *I* is the sum of *N* sleep periods, $\sum_1^N T$. Furthermore, *N* is a geometric random variable with success probability that can be obtained by $1-T^*(\lambda)$; success probability here is the probability of at least one arrival during *T*. The average number of sleep periods is given by $[N]=\frac{1}{1-T^*(\lambda)}$. Therefore, the expected duration of an idle period is given by,

$$E[I]=\frac{E[T]}{1-T^*(\lambda)}. \tag{3}$$

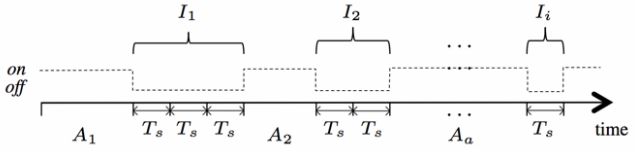


Figure 4. *Active* and idle periods (multiple sleep periods) timing diagram

Substitute (3) into (2) to yield

$$E[A]=\frac{\rho E[T]}{(1-T^*(\lambda))(1-\rho)}. \tag{4}$$

Therefore, the energy saving ratio lower bound which models the fraction of time Tx remains only in the *sleep* state is

$$\breve{R}=\frac{E[I]}{E[I]+E[A]}=(1-\rho). \tag{5}$$

4.2 Upper Bound of Routing Delay

A router managed by ESA with buffer threshold B_{thr} introduces a routing delay denoted by $D(B_{thr})$. The routing delay is defined as the delay experienced by a packet from the instant it is received until it is completely transmitted. Considering the delay in the three states, it is clear that the longest delay is introduced when S_{Tx} is alternating between the *active* and *sleep* states, but not the *rescue* state. This corresponds to a scenario where buffer size and B_{thr} are infinite. Thus, the routing delay upper bound is written as,

$$\hat{D} = \lim_{B_{thr} \to \infty} D(B_{thr}). \quad \textbf{(6)}$$

Based on the queuing system assumptions stated earlier above, and the condition that $B_{thr} \to \infty$, the system can be modeled by a $M^Z/G/1$ queue adopting an exhaustive service and multiple vacations policy (E, MV). Multiple vacations refer to the multiple sleep periods that the system undergoes while the buffer is empty. This class of queues has been well studied in [30]. Based on results in [30], the delay upper bound for a variable sleep timer is

$$\hat{D} = \frac{E[T^2]}{2E[T]} + \frac{\lambda E[Z]E[X^2]}{2(1-\lambda E[X])} + \frac{E[Z^2]E[X]}{2E[Z](1-\lambda E[X])}. \quad \textbf{(7)}$$

However, for a deterministic sleep timer T_s, $\hat{D}$ becomes,

$$\hat{D} = \frac{T_s}{2E[T]} + \frac{\lambda E[Z]E[X^2]}{2(1-\lambda E[X])} + \frac{E[Z^2]E[X]}{2E[Z](1-\lambda E[X])}. \quad \textbf{(8)}$$

4.3 Switching Overhead

Under ESA, the state transition conditions are a function of the traffic load and sleep timer status. Although switching an interface subcomponent *off* conserves energy, frequently switching it back *on* to check occupancy or service arriving packets, consumes more energy than is conserved [15]. As mentioned earlier, an idle period consists of a geometric number of sleep periods denoted by *N*. Thus, for random sleep timer, *T*, the average number of sleep periods in a given idle period is [31].

$$E[N] = \frac{1}{1 - T^*(\lambda)}. \quad \textbf{(9)}$$

However, for a deterministic sleep timer T_s, $E[N]$ becomes

$$E[N] = \frac{1}{1 - e^{(-\lambda T_s)}}. \quad \textbf{(10)}$$

This captures the *on-off* oscillation and switching overhead for ESA.

5 Performance Evaluations

In this section, we study the performance of ESA. We also have comparison to three candidate algorithms that operate based on state transition conditions and sleep timer status and are thus comparable to ESA.

The first algorithm was introduced in [15] by Gupta et. al. and further enhanced by Herrería-Alonso et. al. in [17]. The algorithm measures the inter-arrival time Δ of the last five packets under the assumption that packets arrive following a Poisson process. When buffer occupancy *b* drops to zero, the sleep time is calculated such that the probability that a packet arrives during the sleep period is less than 0.1. If the evaluated sleep time is less than the transition time $T_t^{a \to s}$, the transition is ignored and the interface remains in the *active* state. Alternatively, the interface switches to the *sleep* state. We refer to this algorithm as Dynamic Sleep timer (DS) algorithm.

The other two candidate algorithms, frame transmission (FT) and burst transmission (BT), were presented in [18]. Both algorithms switch the interface to the *sleep* state when the buffer is empty for a predefined, deterministic sleep period. However, FT wakes up the interface upon the first packet arrival, whereas BT waits until the buffer occupancy *b* exceeds a threshold B_w. Despite their simplicity, FT and BT have shown significant energy savings [18].

Unlike ESA, none of the three candidate algorithms provides a mechanism to avoid packet loss for high peak-to-average rate traffic. Table 1 tabulates ESA's and the candidate algorithms' transition conditions. To avoid backlog delays, all algorithms can switch to the *sleep* state only when all queued packets have been transmitted. Furthermore, all algorithms use a constant sleep timer, except DS, which computes T_s based on the packet inter-arrival times. Compared with the considered algorithms, ESA does not switch to *active* state until the sleep timer expires. This introduces a higher energy saving margin; however, it also puts packets at a higher risk of being dropped due to buffer overflow. To rescue packets from loss, the *rescue* state is activated when buffer occupancy *b* reaches B_{thr}.

Table 1. Comparison table of ESA's and candidate algorithms' transition conditions

Algorithm	*sleep*	*active*	*rescue*
DS	$b=0$ and $T_s > T_t^{s \to a}$ $T_s \mid Probability\ (\Delta < T_s) < 0.1$	$B > 0$	NA
FT	$b=0$	$B > 0$	NA
BT	$b=0$	$B > B_w$	NA
ESA	$b=0$	T_s expires	$b > B_{thr}$

The performance study is based on the NS-2 simulation experiments. To get insights into the performance of ESA under a close to network setting

in real world, we use real traffic traces and practical simulation parameters to setup simulation experiments. A traffic trace is collected at the Indianapolis router node on the Internet2 consortium network [32]. The trace is collected on a backbone link to Cleveland, OH with an average load of $\rho = 0.4135$. The average packet size is 1000 bytes. The duration of the trace file is 10 minutes; the file is publicly available at Waikato Internet Traffic Storage [33]. Replaying real traces in NS-2 simulations replicates real Internet links, resulting in more realistic evaluations. In our evaluations, we refer to the first trace as "Average Load", and we refer to the combination of the trace and a duplicate of it, resulting in an average load of $\rho = 0.827$ as "High Load". Our 1000BASE-T physical layer parameters are also realistic, whose details can be found in the IEEE 802.3 standard [19]. A summary of the simulation parameters is given in Table 2. Note that we considered a range of buffer sizes, $B \in \{20, \cdots, 500\}$.

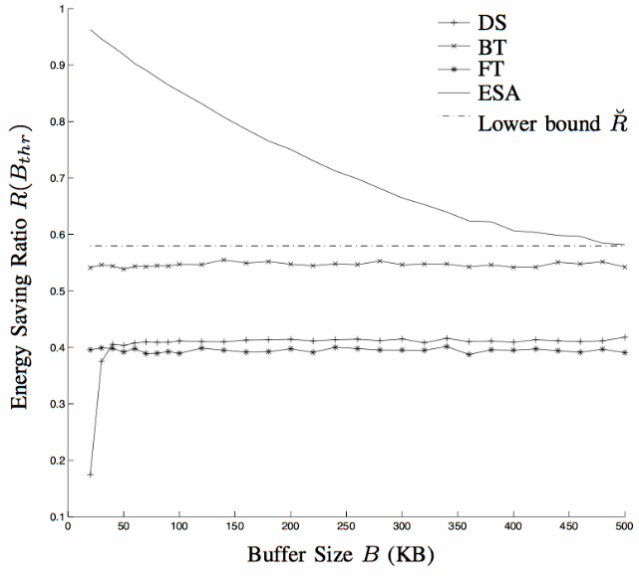
Table 2. Simulation parameters

Parameter	value
T_s	2.5 millisecond
$T_t^{a \to s}$	2.88 microsecond
$T_t^{s \to a}$	4.88 microsecond
B_w	0.5 B
B_{thr}	0.5 B

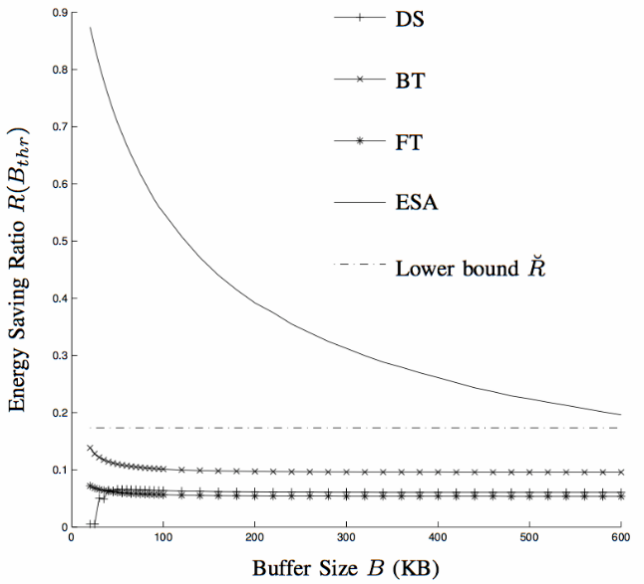
Utilizing the buffer management class `ESQueue` developed in Appendix A, we focus on the performance of ESA at a single link. The simulated network links have 1 Gbps capacities and are configured with the parameters tabulated above. The traffic file was captured at a real backbone router and resembles the traffic of a large scale network. The average batch size and batch arrival rate varies among the files. Trace files are attached to a User Datagram Protocol (UDP) agent modeling the simulated flow source. For each trial, the simulation experiments for a given set of parameters started at random points in the trace file and run for 200 seconds. Each experiment is repeated for twenty trials, and the average of the measured parameter is computed over all trials. With the interface switched to the *rescue* state, the dynamic routing available in NS-2 and Multiprotocol Label Switching (MPLS) are used to find alternative paths to route the packets arriving over the remaining time of T_s. Our evaluations focus on the algorithms' performances over a practical buffer size range: 80 KB to 256 KB [15, 35].

5.1 Energy Saving

Figure 5 compares the performance of ESA against those of DS, FT, and BT in terms of energy efficiency. Moreover, it validates ESA's derived energy saving ratio lower bound for real traffic traces. The energy saving ratio captures the ratio of the time spent in either the sleep state, rescue state, or switching among states to the total operating time. ESA is shown to be superior to all other algorithms. Under an average load, the improvement, over the typical range of buffer sizes, is in the range of 15% to 35% relative to BT. In comparison to DS and FT, ESA has a higher range of improvement, from 30% to 50%. Under high load, ESA continues to outperform BT and the other algorithms by approximately 25% to 50%, as shown in Figure 5.



(a) Average Load



(b) High load

Figure 5. Energy saving ratio versus buffer size for DS, FT, BT and ESA under average and high traffic loads

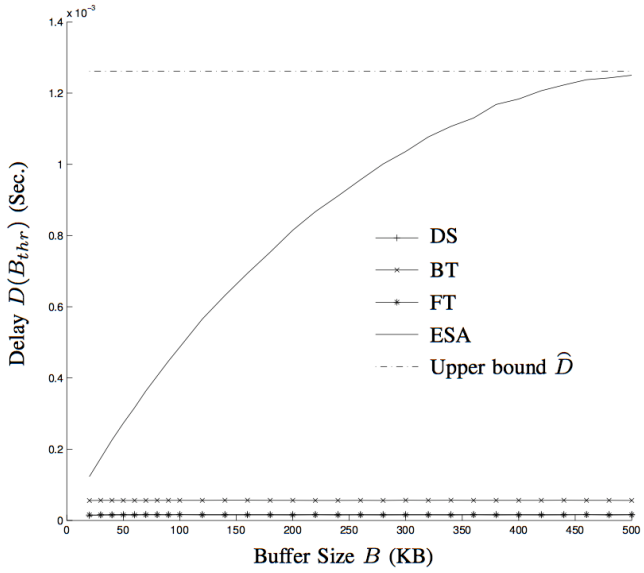
The worst performance is observed for FT. This algorithm switches to the *active* state upon packet arrivals, which does not allow for long sleep periods and thus achieves less energy saving. A slight improvement over FT is observed for DS that also switches to *active* state on packet arrival but dynamically adjusts T_s based on the packets' inter-arrival times. This slight improvement is computationally expensive because DS implements a root finding algorithm in every sleep period. Compared with DS and FT, packets coalescing in BT brings energy savings of 15% under the average load and only of 5% under high load. The sleep persistent approach adopted in ESA is shown to be the most energy efficient. Even the least energy saving ratio for ESA, as represented by $\bar{R}$, is higher than the $R(B_{thr})$ for DS, BT, and FT.

5.2 Packet Delay

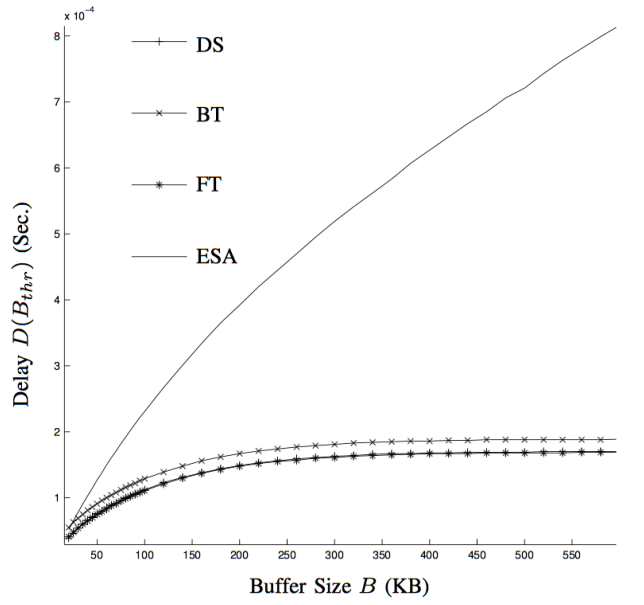
As shown in Figure 6, under average load the gap in packet delay for ESA and BT is as high as a factor of 9 over the practical range 80 KB to 256 KB. This is because ESA does not begin to serve packets until the sleep timer expires. Conversely, DS and FT begin to serve packets as soon as a single packet arrives, thus significantly reducing packet delay. The slight increase in delay for BT relative to DS and FT is due to the time it waits until B_w packets accumulate before it starts to serve packets. On the other axis, for high load, the gap is reduced to a factor of 3. Figure 6a shows that although packets experience a longer delay under ESA, the delay upper bound $\hat{D}$, i.e., the worst-case scenario, is slightly greater than half that of T_s. This finding implies that in the worst case, the delay is less than the maximum allowable delay configured by the sleep timer T_s. An interesting observation here is that ESA takes advantage of the delay margin to conserve energy without exceeding T_s. It is clear from Figure 6(a) and Figure 6(b) that the delay under high load is less than the delay under average load. This occurs because under high load the buffer threshold B_{thr} is reached faster and packets wait a shorter time before the interface switches to the *rescue* state.

5.3 Transition Time

The ratio of total transition time among states to the simulation length is shown in Figure 7. In addition to the energy wasted in switching to another state, frequent oscillation among states increases packet delay because the interface is not serving packets. Therefore, reducing the transit time is desirable. The plots shown in Figure 7 demonstrate that ESA again outperforms DS, BT, and FT under both average and high loads.



(a) Average load

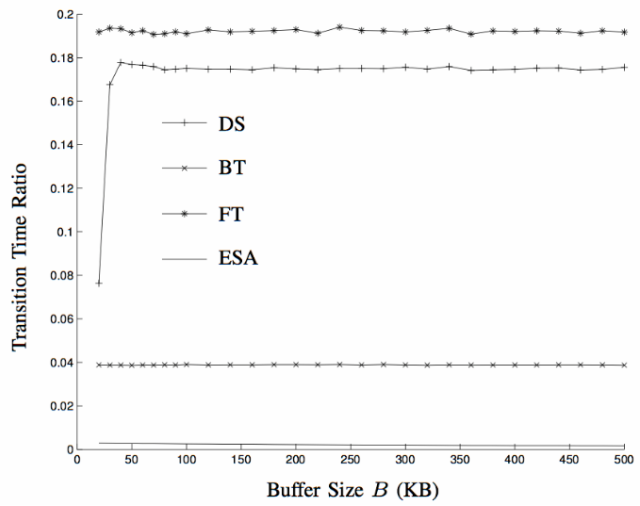


(b) High load

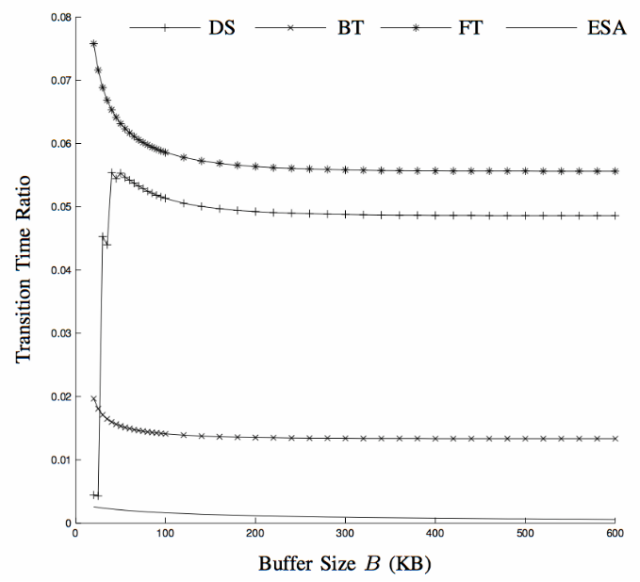
Figure 6. Packet delay versus buffer size for DS, FT, BT and ESA under average delay and high traffic loads

5.4 Packet Loss

Evaluating the packet loss ratio is critical for energy management algorithms because long sleep periods may lead to buffer overflow and packet loss. Figure 8 shows the decay in packet loss versus buffer size for all algorithms. Under both loads, Figure 8(a) and Figure 8(b) show that ESA is the winner in terms of packet loss, despite its sleep persistence. This is due to the *rescue* state, in which the algorithm rescues packets before the buffer becomes full and packets are dropped.

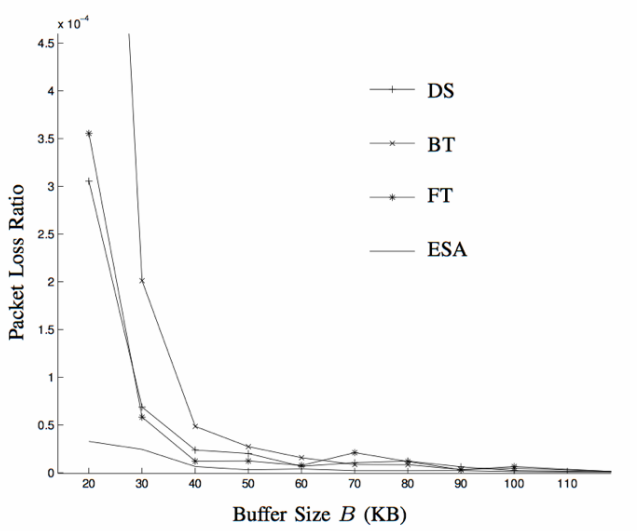


(a) Average load

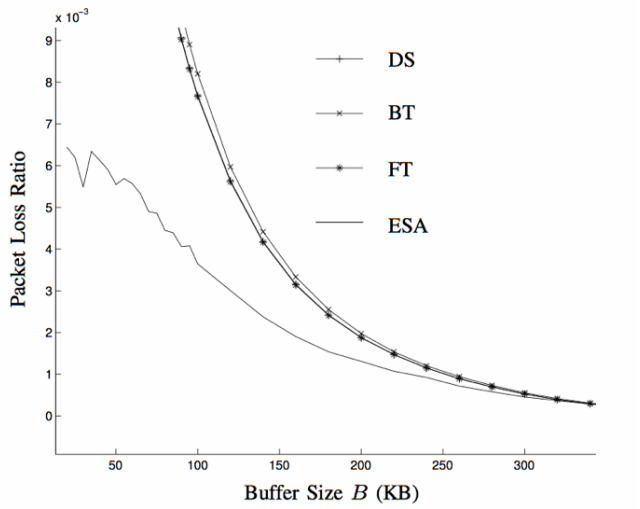


(b) High load

Figure 7. ESA's and candidate algorithms' ratio of transition time to operating time



(a) Average load



(b) High load

Figure 8. Packet loss ratio for ESA and candidate algorithms over a range of buffer sizes

6 Conclusions

In this paper, a distributed and load adaptive energy management algorithm was proposed for Ethernet networks. The algorithm switches the interface to the *sleep* state until either the sleep timer expires or the buffer occupancy reaches a buffer threshold. In the former case, the interface switches to the *active* state to serve packets; however, in the latter case, the interface switches to the *rescue* state to prevent packets from being dropped. Although the proposed algorithm is sleep persistent, the introduced packet delay is bounded by half of the sleep timer. An NS-2 based simulation model is developed for the proposed algorithm. Furthermore, numerical evaluations verified the accuracy of the simulation model against derived bounds and metrics. NS-2 experiments with practical setups and real traffic traces have shown that the proposed algorithm is superior to other considered candidate algorithms.

Acknowledgment

The authors thank Dr. Sergio Herrería-Alonso with the Department of Telematics Engineering at the University of Vigo, Spain for valuable discussions on algorithms the presented in [17]. Also, we would like to thank him for sharing the simulation source code with us.

The project was partially funded by the Kuwait Foundation for the Advancement of Sciences under project code: P314- 35EO-01. Phone Lin's work was supported in part by Ministry of Science and

Technology (MOST), R.O.C., under grant numbers MOST 105-2622-8-009-008, MOST 104-2622-8-009-001-, MOST 103-2221-E-002-249-MY3, MOST 103-2221-E-002-152-MY3, MOST 104-2923-E-002-005-MY3, MOST 105-2627-E-002-00, MOST 104-2627-E-002-003, MOST 105-2218-E-002-013-, and MOST 104-3115-E-002-004-, and by Chunghwa Telecom.

References

[1] The Climate Group on behalf of the Global eSustainability Initiative (GeSI), *Smart 2020: Enabling the Low Carbon Economy in the Information Age*, http://www.gesi.org.

[2] Global Action Plan, *An Inefficient Truth*, http://globalactionplan.org.uk.

[3] R. Bolla, R. Bruschi, F. Davoli, F. Cucchietti, Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures, *IEEE Communications Surveys & Tutorials*, Vol. 13, No. 2, pp. 223-244, June, 2011.

[4] R. Bolla, R. Bruschi, A. Carrega, F. Davoli, Green Networking with Packet Processing Engines: Modeling and Optimization, *IEEE/ACM Transactions on Networking*, Vol. 22, No. 1, pp. 110-123, February, 2014.

[5] L. Chiaraviglio, M. Mellia, F. Neri, Minimizing ISP Network Energy Cost: Formulation and Solutions, *IEEE/ACM Transactions on Networking*, Vol. 20, No. 2, pp. 463-476, April, 2012.

[6] Q. Li, M. Xu, Y. Yang, L. Gao, Y. Cui, J. Wu, Safe and Practical Energy-Efficient Detour Routing in IP Networks, *IEEE/ACM Transactions on Networking*, Vol. 22, No. 6, pp. 1925-1937, December, 2014.

[7] M. Andrews, S. Antonakopoulos, L. Zhang, Rate-Adaptive Scheduling Policies for Network Stability and Energy Efficiency, *IEEE/ACM Transactions on Networking*, Vol. 23, No. 6, pp. 1755-1764, December, 2015.

[8] M. Caria, F. Carpio, A. Jukan, M. Hoffmann, Migration to Energy Efficient Routers: Where to Start?, *IEEE International Conference on Communications*, Sydney, NSW, Australia, 2014, pp. 4300-4306.

[9] A. Coiro, M. Polverini, A. Cianfrani, M. Listanti, Energy Saving Improvements in IP Networks through Table Lookup Bypass in Router Line Cards, *International Conference on Computing, Networking and Communications*, San Diego, CA, 2013, pp. 560-566.

[10] M. Polverini, A. Cianfrani, A. Coiro, M. Listanti, R. Bruschi, Freezing Forwarding Functionality to Make the Network Greener, *Computer Networks*, Vol. 78, pp. 26-41, February, 2015.

[11] M. Gupta, S. Singh, Greening of the Internet, *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, Karlsruhe, Germany, 2003, pp. 19-26.

[12] M. Gupta, S. Grover, S. Singh, A Feasibility Study for Power Management in LAN Switches, *Proceedings of the 12th IEEE International Conference on Network Protocols,* Berlin, Germany, 2004, pp. 361-371.

[13] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, J. A. Maestro, IEEE 802.3az: The Road to Energy Efficient Ethernet, *IEEE Communications Magazine*, Vol. 48, No. 11, pp. 50-56, November, 2010.

[14] M. Gupta, S. Singh, Dynamic Ethernet Link Shutdown for Energy Conservation on Ethernet Links, *IEEE International Conference on Communications,* Glasgow, UK, 2007, pp. 6156-6161.

[15] M. Gupta, S. Singh, Using Low-power Modes for Energy Conservation in Ethernet LANs, *26th IEEE International Conference on Computer Communications,* Barcelona, Spain, 2007, pp. 2451-2455.

[16] G. Ananthanarayanan, R. H. Katz, Greening the Switch, in *Proceedings of the 2008 Conference on Power Aware Computing and Systems*, San Diego, CA, 2008, pp. 7-7.

[17] S. Herrería-Alonso, M. Rodríguez-Pérez, M. Fernández-Veiga, C. López-García, Opportunistic Power Saving Algorithms for Ethernet Devices, *Computer Networks*, Vol. 55, No. 9, pp. 2051-2064, June, 2011.

[18] S. Herrería-Alonso, M. Rodríguez-Pérez, M. Fernández-Veiga, C. López-García Optimal Configuration of Energy-efficient Ethernet, *Computer Networks*, Vol. 56, No. 10, pp. 2456-2467, July, 2012.

[19] IEEE Computer Society, *IEEE Standard for Ethernet, IEEE Std 802.3-2012 (Revision to IEEE Std 802.3-2008)*, December, 2012.

[20] S. Vitturi, F. Tramarin, Energy Efficient Ethernet for Real-time Industrial Networks, *IEEE Transactions on Automation Science and Engineering*, Vol. 12, No. 1, pp. 228-237, January, 2015.

[21] F. Tramarin, S. Vitturi, Strategies and Services for Energy Efficiency in Real-time Ethernet Networks, *IEEE Transactions on Industrial Informatics*, Vol. 11, No. 3, pp. 841-852, June, 2015.

[22] A. Chatzipapas, V. Mancuso, An M/G/1 Model for Gigabit Energy Efficient Ethernet Links with Coalescing and Real-trace-based Evaluation, *IEEE/ACM Transactions on Networking*, Vol. 24, No. 5, pp. 2663-2675, October, 2016.

[23] M. Rodríguez-Pérez, M. Fernández-Veiga, S. Herrería-Alonso, M. Hmila, C. López-García, Optimum Traffic Allocation in Bundled Energy-efficient Ethernet Links, *IEEE Systems Journal*, Vol.12, No. 1, pp. 593-603, March, 2018.

[24] J. Moy, *OSPF Version 2*, RFC 2328, April, 1998.

[25] C. Hedrick, *Routing Information Protocol*, RFC 1058, June, 1988.

[26] Y. Rekhter, T. Li, *A Border Gateway Protocol 4 (BGP-4)*, RFC 1771, March, 1995.

[27] J. Aweya, IP Router Architectures: An Overview, *International Journal of Communication Systems*, Vol. 14, No. 5, pp. 447-475, June, 2001.

[28] A. Cianfrani, V. Eramo, M. Listanti, M. Marazza, E. Vittorini, An Energy Saving Routing Algorithm for a Green OSPF Protocol, *IEEE INFOCOM Conference on Computer Communications Workshops*, San Diego, CA, 2010, pp. 1-5.

[29] L. Chiaraviglio, M. Mellia, F. Neri, Energy-aware Backbone

Networks: A Case Study, *IEEE International Conference on Communications Workshops*, Dresden, Germany, 2009, pp.1-5.

[30] N. Tian, G. Zhang, *Vacation Queueing Models: Theory and Applications*, Springer-Verlag, 2006.

[31] A. Borthakur, G. Choudhury, On a Batch Arrival Poisson Queue with Generalized Vacation, *Sankhya: The Indian Journal of Statistics, Series B*, Vol. 59, No. 3, pp. 369-383, December, 1997.

[32] Internet2, *Internet2 consortium*, http://www.internet2.edu/about-us/.

[33] WAND Network Research Group, *Waikato Internet Traffic Storage Project*, http://wand.net.nz.

[34] T. Issariyakul, E. Hossain, *Introduction to Network Simulator NS2*, Springer Science & Business Media, 2011.

[35] Cisco Systems, *Ethernet Switches Specifications*, http://www.cisco.com.

Biographies

Mohamad Khattar Awad, earned the B.A.Sc. in electrical and computer engineering (communications option) from the University of Windsor, Ontario, Canada, in 2004 and the M.A.Sc. and Ph.D. in electrical and computer engineering from the University of Waterloo, Ontario, Canada, in 2006 and 2009, respectively.

From 2004 to 2009 he was a research assistant in the Broadband Communications Research Group (BBCR), University of Waterloo. In 2009 to 2012, he was an Assistant Professor of Electrical and Computer Engineering at the American University of Kuwait. Since 2012, he has been with Kuwait University as an Assistant Professor of Computer Engineering.

Dr. Awad's research interest includes wireless and wired communications, software-defined networks resource allocation, wireless networks resource allocation, and acoustic vector-sensor signal processing. He received the Ontario Research & Development Challenge Fund Bell Scholarship in 2008 and 2009, the University of Waterloo Graduate Scholarship in 2009, and a fellowship award from the Dartmouth College, Hanover, NH in 2011. In 2015, he received the Kuwait University Teaching Excellence Award.

Phone Lin is a Professor at National Taiwan University (NTU), holding professorship in the Department of Computer Science and Information Engineering, Graduate Institute of Networking and Multimedia, Telecommunications Research Center of College of EECS, and Graduate Institute of Medical Device and Imaging, Collage of Medicine.

Lin serves on the editorial boards of IEEE Trans. on Vehicular Technology, IEEE Wireless Communications Magazine, IEEE Network Magazine, IEEE Internet of Things Journal, etc. He has also been involved in several prestigious conferences, such as holding, Local Arrangement Co-Chair, IEEE VTC2010-Spring, Taipei, Taiwan, the Technical Program Chair of WPMC 2012, Co-Chair of the Wireless Networking Symposium of IEEE Globecom 2014, and TPC member of Infocom 2010-2014. He was Chair of IEEE Vehicular Technology Society Taipei Chapter 2014-2015.

Lin served as Associate Chairman, Department of Computer Science and Information Engineering, National Taiwan University from 2010-2011, and as Director in Computer and Information Networking Center at National Taiwan University from 2005-2009.

Dr. Lin's current research interests include "Fog Networks," "Machine to Machine (M2M)/Internet of Things (IoT)," "Software Defined Networks (SDN)," "Smart Data Pricing," "Relation, Privacy, Security of Social Networks," "Decision Process" and "Performance Modeling." Lin has received many prestigious technical research/service awards, such as The Outstanding Research Award, Minstry of Science and Technology, Taiwan in 2016, The Best Young Researcher of IEEE ComSoc Asia-Pacific Young Researcher Award in 2007, The Distinguished Electrical Engineering Professor Award of the Chinese Institute of Electrical Engineering in 2012, Ten Outstanding Young Persons Award of Taiwan (Science & Technology) in 2009.

Gi-Ren Liu received his Ph.D. degree from National Taiwan University, Taiwan, ROC in 2013. In 2013, he joined the Department of Computer Science, National Chiao Tung University, Taiwan, as a postdoctoral fellow. In 2014, he was a visiting scholar at University of California, Davis. Now, he is an assistant professor at the Department of Mathematics, National Cheng Kung University (NCKU), Taiwan. His current research interests include spectral analysis, queuing theory, Markov decision processes and their applications on personal communications services and smart grids.

Appendix A: NS-2-based Simulation Model

In this Appendix, we describe the simulation model based on ESA in NS-2. We introduce new classes to the NS-2 hierarchy of C++ classes. Figure 9 shows the relationship between the new classes and the C++ standard classes from which they are derived.

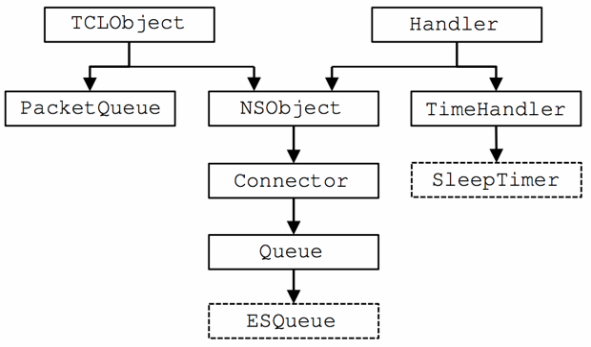


Figure 9. Part of the NS-2 C++ classes hierarchy [34]. New classes are shown in boxes with dashed lines

The new class, `ESQueue` is derived from `Queue`. Class `Queue` simulates the queue interface, which implements two functions, including buffer management and packet scheduling [34]. Buffer management regulates the buffer occupancy of the queue, and packet scheduling admits and drops packets. `Queue` also manages the buffer through the object `PacketQueue` which simulates low-level operations of the buffer.

The constructor of class `ESQueue` creates a pointer `* q_` to object `PacketQueue`. Furthermore, it sets `pq_`, which is derived from `Queue`, equal to `q_`. The `PacketQueue` maintains a linked list of packets and consists of three functions: `enque(p)`, `deque()` and `length()`. Whereas function `enque(p)` adds a packet `* p` to the tail of the buffer, function `deque()` returns a pointer to the packet at the head of the buffer. Function `length(p)` returns the current length of the buffer.

The `ESQueue` maintains an enumerated list of states: `active`, `sleep`, `rescue`, `trans-to-sleep`, `trans-to-active`. The state is stored in the variable `state_`. The first three states represent the states described in Section 3. The last two states, `trans-to-sleep` and `trans-to-active`, model the transition delay among states. Transition states are introduced to model circuit state switching delays. In other words, transition states set the timer to either $T_t^{a\to s}$ or $T_t^{s\to a}$ and switches to either `sleep` or `active` on timer expiry, respectively. Furthermore, `ESQueue` implements two pure virtual functions of `Queue`, `enque(p)` and `deque()`, which places and takes packets from `* q_`, respectively. The pseudo-codes are shown in Functions 1 and 2.

Function 1 Pseudo-code of `ESQueue::enque(p)`

```
1: if adding packet p results in overflow then
2:     drop packet p
3: else
4:     q_->enque(p)
5: end if
6: if state_==sleep & q_->length()+1 > B_thr then
7:     switch_to_(rescue)
8: end if
```

Function 2 Pseudo-code of `ESQueue::deque( )`

```
1: if state_ != active then
2:     return 0
3: else // the interface is active
4:     if q_->length() == 0 then // buffer empty
5:         switch_to_(trans-to-sleep)
6:         return 0
7:     else
8:         return q_->deque( )
9:     end if
10: end if
```

Functions 1 and 2 call the `switch_to_(state_)` to handle switching among states, setting the sleep timer T_s, and setting transition timers $T_t^{a\to s}$ or $T_t^{s\to a}$. The timers are implemented in the class `SleepTimer` that is a child class of `TimerHandler` as shown in Figure 9. In `TimerHandler`, we implement the function `resched(delay_)` to restart the timer and place a timer expiration event after `delay_` seconds. In `SleepTimer`, we implement a pure virtual function `expire(Event *)` for the expiry of timers. Function 3 shows the pseudo codes of `switch_to_(state_)`, where `st_` is an object of `SleepTimer`.

Function 3
Pseudo-code of `ESQueue::switch_to_(state_)`

```
1: if state_ == sleep then
2:     st_.resched(T_s) //set timer for sleep state
3: else if state_ == trans-to-sleep then
4:     st_.resched(T_t^{a→s}) //set timer for the trans-to-
       sleep state
5: else if state_ == trans-to-active then
6:     st_.resched(T_t^{s→a}) //set timer for the trans-to-
       active state
7: else if state_ == rescue then
8:     st_.resched(remaining time in T_s ) //set
       timer for the rescue state
9:     re-route packets
10: else
11:     state_ must be active and timer is not set
12: end if
```

In Function 3, when a timer expires, the `expire(Event *)` event is triggered. Depending on the state when the timer expires, different action is taken. If the `expire(Event *)` function is invoked when the interface is in the `sleep` or `trans-to-`

`sleep` state, and the buffer is empty, the interface switches to the `trans-to-active` state. If the buffer is empty, the interface remains in the `sleep` state. Alternatively, if the function `expire(Event *)` is invoked while the interface is in the `rescue` state, the interface switches to the `trans-to-active` state and re-routes packets. Similarly, if the timer fires while the interface is in the `trans-to-active` state, it switches to the `active` state. Note that the interface does not switch between the `active` state and the `sleep` state without going through transition states. Function 4 shows the pseudo codes of `expire(Event *)` where `SleepTimer` creates a pointer, `ESq_`, to `ESQueue`.

Function 4
Pseudo-code of `SleepTimer::expire(Event *)`

```
1: if ESq_->state_==sleep or
   ESq_->state_==tran-to-sleep then
2:     if buffer occupancy > 0 then
3:         ESq_->switch_to_(trans-to-active)
4:     else
5:         ESq_->switch_to_(sleep)
6:     end if
7: else if ESq_->state_==rescue then
8:     ESq_->switch_to_(trans-to-active)
9:     re-route packets
10: else //state_ must be trans-to-active
11:     ESq_->switch_to_(active)
12: end if
```

The buffer class `ESQueue` simulates ESA buffer management discipline for a single link in NS-2. Thus, it can be integrated in a topology of any size with either of the available NS-2 link types simplex-link or duplex-link. The proposed simulation model can be easily extended (i.e., by modifying the transition conditions and states) to simulate other buffer management algorithms without modifying the architecture.

In Appendix B, we verify the simulation model presented in this Appendix against analytical models derived in Section 4. Results in Appendix B confirm the accuracy of the simulation model in modeling the proposed algorithm. The fundamental difference between model validation and performance evaluation experiments presented in Section 5 is the type of traffic fed to the simulator. In the simulation model verification experiments, traffic source files are generated in Matlab with properties matching the assumptions stated in section 4. The traffic fed to the simulator in the performance evaluation experiments (Section 5), however, is captured at a node in a real network. This traffic is available on the Internet2 consortium network [32].

Appendix B: Simulation Model Verification

We verify the simulation model against bounds and metrics derived in Section 4. The length of each experiment is set to 200 seconds during which packets arrive following a compound (batch) Poisson process and with a uniform batch size (1000 bytes on average). For the purpose of verification, we simulate a single duplex link implementing ESA as its buffer management discipline. The capacity of the simulated link was 1 Gbps and the arrival rate was varied to have a utilization factor ρ in the range 0.1 to 0.5. The traffic trace files were generated and converted to binary files in Matlab before being attached to the source node of the simulated link. Experiments were setup based on the 1000BASE-T physical layer characteristics proposed in the IEEE 802.3 Energy Efficient Ethernet (EEE) [19]. Specifically, the standard transition times $T_t^{a\to s}$ and $T_t^{s\to a}$ were set to 2.88 microseconds and 4.88 microseconds, respectively. In addition, the sleep timer was configured to have T_s = 2.5 milliseconds, a common value in the literature [15, 17-18].

Over a range of buffer sizes (20 KB to 320 KB) and various link utilization values, the model is verified in terms of accuracy of the switching overhead and the tightness of the bounds. It is worth noting that the current practical buffer size for a single port in a switch ranges from 80 KB to 256KB [15, 35]; however, we use a larger range to assess the accuracy of the model under both low and high buffer occupancy conditions. Figure 10 shows the energy saving ratio $R(B_{thr})$ for threshold B_{thr} = 0.5B, where $B \in \{20, \cdots, 500\}$. The tightness of the lower bound for all considered values of link utilization, ρ, validates the simulation model. It is also clear that ESA adapts to the traffic arrival rate: for one tenth decrease in link utilization, $R(B_{thr})$ increases by approximately 10%.

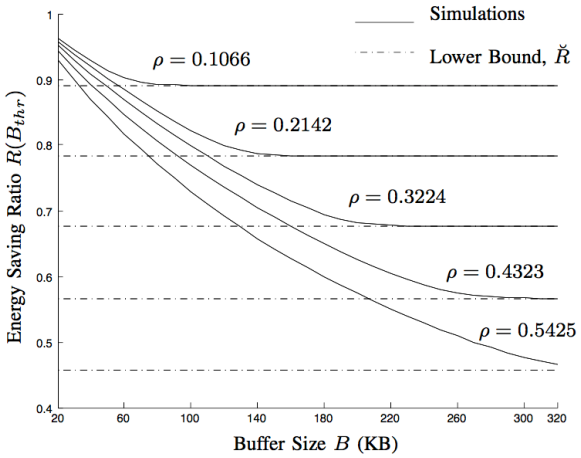


Figure 10. Energy Saving Ratio versus buffer size for a range of traffic intensities

Figure 11 shows the average packet delay $D(B_{thr})$ and the derived delay upper bound $\hat{D}$. The plots demonstrate that the derived upper bound tightly binds the packet delay for all considered values of ρ, which supports the accuracy of the simulation model. The results reflect the negligible effect of traffic characteristics on the upper bound of the delay compared with $T_s/2$. This is evident from the overlap of $\hat{D}$ at $T_s/2$ = 1.25 milliseconds for a wide range of ρ and B. Conversely, Figure 11 shows lower average delays for higher link utilization. This is because B_{thr} is reached at faster rate for higher traffic intensity and hence the interface switches to *rescue* state and remains in it for a larger portion of T_s. In other words, more packets are served by alternative paths with lower delays, leading to a lower average delay. An interesting conclusion is that the delay is bounded by approximately half of the sleep timer. Thus, given the maximum tolerable delay, the sleep timer can be easily set.

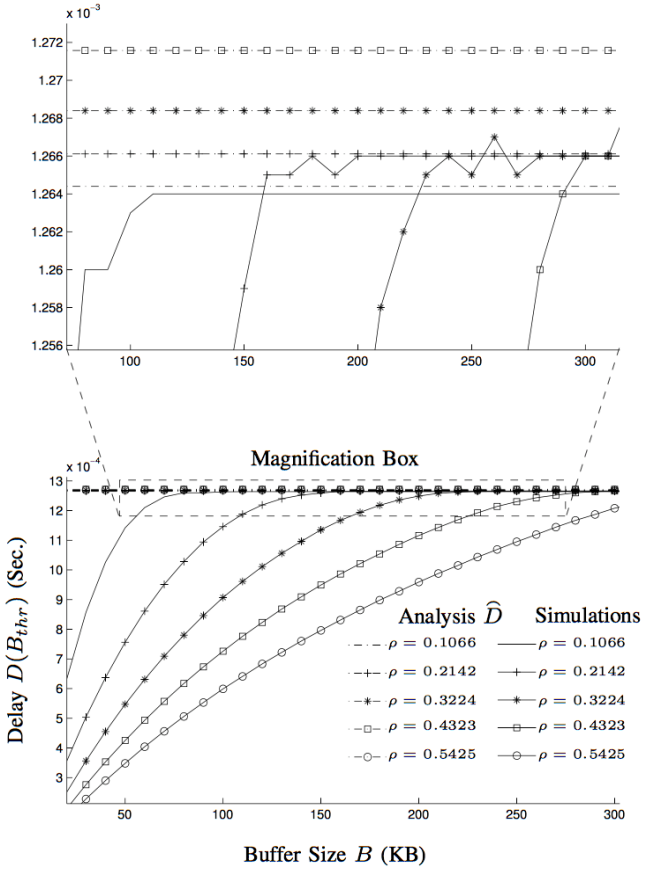


Figure 11. Simulated and derived packet delay versus buffer size

A larger value of T_s minimizes energy consumption for two reasons. First, the interface remains in the *off* state longer and higher energy savings are thus achieved. Second, the interface goes through fewer *on-off* oscillations, which also translates to lower energy consumption. Conversely, a larger T_s introduces longer delays to packets that arrive during the *off* state. Figure 12 plots the average number of sleep periods in an idle period, $E[N]$. The presented graphs depict $E[N]$ for a range of packet arrival rates and two commonly used values for T_s. It can be observed simulations matches analysis with high accuracy. It is also clear that under ESA, the interface experiences low *on-off* oscillations for reasonable arrival rates; for higher arrival rates, E[N] converges to 1. Hence, the following conclusion can be drawn: to balance between large delays and *on-off* oscillations, T_s should be sufficiently large to fill the packets' inter-arrival time with fewest sleep periods without exceeding the maximum tolerable delay.

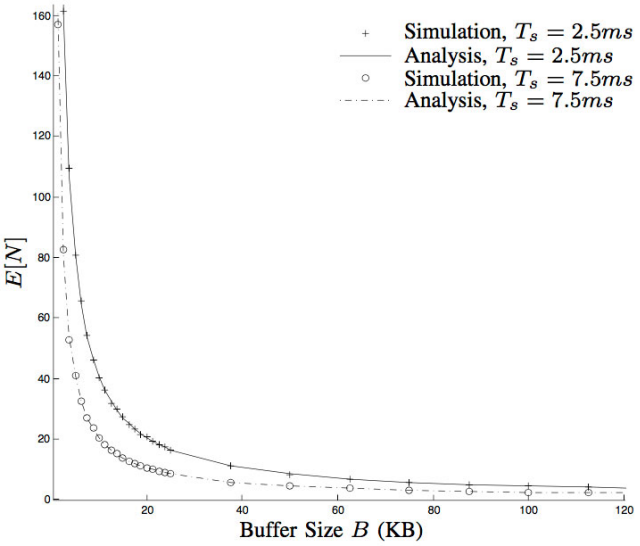


Figure 12. Validation of the simulation model in terms of the average number of sleep periods