Trading off Power Consumption and Delay in Packet Forwarding Engines with Adjustable Service Rate

Raffaele Bolla[1,2], Roberto Bruschi[2], Franco Davoli[1,2], and Paolo Lago[1]

1. DITEN, University of Genoa
Genoa, Italy
{name.surname@unige.it}

2. CNIT
Genoa, Italy
{name.surname@cnit.it}

***Abstract*— This paper presents an Adaptive Rate (AR) control policy for trading off power consumption and performance of packet forwarding engines. We assume a Forwarding Engine (FE) whose power can be scaled through *Dynamic Voltage and Frequency Scaling* (DVFS) and *Clock Gating* (CG). In a typical architecture, data packets are always transferred at the maximum speed consuming the maximum power. We proposed to tradeoff energy efficiency and forwarding performance of the FE by acting on the power-delay product. To this purpose, we provide an optimization framework for dynamically adjusting the capacity of single server queuing system. Then, we design a practical AR policy, which can be easily extended for multi-core/multi-processor architectures. The performance evaluation revealed that a significant power reduction is experienced when using our policy, with a small increase of packet delays during low traffic condition and without compromising performance during peak hours.**

***Keywords*— Green Networking; DVFS; Clock Gating**

I. INTRODUCTION

Energy management in network devices is becoming a crucial issue. Despite obvious environmental reasons, this interest springs from heavy and critical economical needs, since both energy cost and network electrical requirements have shown a continuous growth with an alarming trend over the past years [1]. It is well known that network devices are dimensioned according to peaks in the load. As a consequence, the more the network capacity, the more would be the energy required to supply devices. It has been observed that communication networks are mainly under-utilized during normal operations, leaving a large room for energy savings [2]. However, the power consumption of current networking equipment remains more or less constant even in the presence of fluctuating traffic loads. This situation suggests the possibility of adapting network capacity and energy requirements according to the actual traffic profiles [3].

As regards the breakdown of energy consumption among the device parts, Tucker et al. [4] focused on high-end router platforms, and decomposed the energy requirements on the basis of network-specific functionalities. They estimated that internal packet forwarding engines represent one of the most energy-hungry components in many network devices and require about 32% of the overall energy.

Starting from these considerations, we faced the possibility of introducing power saving strategies in forwarding engines. In detail, we assume the FE can save power through dynamic voltage and frequency scaling and clock gating.

We aim to improve the energy efficiency of the FE without compromising its forwarding performance by exploiting an on-line adaptive rate policy. Our approach is to tradeoff power consumption and forwarding performance by acting on the power-delay product. To this purpose, in Section II we provide an optimization framework for dynamically adjusting the processor performance. For the sake of simplicity we considered a single server queue system with the aforementioned power saving capabilities. Then, in Section III, we propose a practical adaptive rate policy, which can be easily extended for multi-core/multi-processor systems. Finally, Section IV and Section V report the performance evaluation and conclusions, respectively.

II. OPTIMIZATION FRAMEWORK

A. The Power Consumption Model

Most of the power saving techniques currently studied by the research community already exist in general purpose processors. The most common power saving technologies are DVFS and CG. DVFS allows dynamically modulating the capacity of a processor by tuning the clock frequency and supply voltage, while CG exploits short inactivity periods to disable portions of the circuitry.

Assuming the FE supports DVFS and CG mechanisms, we can derive an analytical power consumption model of the system.

The power consumption of a CMOS based processor arises from two main contributions: a static and a dynamic one.

$$\Phi = \Phi_{\text{stat}} + \Phi_{\text{dyn}} \quad (1)$$

The static power dissipation, Φ_{stat}, mainly depends on the CMOS size, the silicon operating temperature and the supply voltage V_x, and it can contribute to the 42%-50% of the total power in 90-65 nm processor, respectively [5]. Because of the complex relationship between V_x and Φ_{stat} and its dependency to circuit-level features, we assumed it to be constant.

The second term, Φ_{dyn}, derives from the charging and discharging of the total processor's capacitance during operations, and it somehow represents the “ideal” power absorption of the circuit, since it is due to the real transitions of

978-0-9836283-9-2

CMOS logical states [6]. In more detail, Φ_{dyn} can be expressed as:

$$\Phi_{dyn} = \nu\, C\, V_x^2 f_x \quad (2)$$

where ν is the fraction of gates actively switching, C is the total transistor gate capacitance of the entire module, V_x is the supply voltage, and f_x is the clock frequency.

As previously sketched, DVFS reduces power by scaling the operating clock frequency and supply voltage. In detail, the implementation of DVFS is generally performed by pre-selecting a set of feasible and stable couple of V_x and f_x.

In a DVFS-capable processor, V_x raised to some power γ and f_x are proportional, $f_x \propto V_x^{\gamma}$ (with $0 < \gamma \leq 1$) [7]; this implies $V_x \propto f_x^{1/\gamma}$. Moreover, the working frequency is linearly related with the packet service rate of the FE:

$$f_x = \beta\, \mu_x \quad (3)$$

where β is the number of cycles needed to forward a packet and can be assumed constant. μ_x is the packet service rate of the engine when working at the clock frequency f_x.

Given the relationship between the supply voltage and the working frequency and by substituting (3) in (2), the dynamic power consumption turns out to be:

$$\Phi_{dyn}(\mu_x) = K\, \mu_x^{(1+\frac{2}{\gamma})} \quad (4)$$

where K is a hardware-dependent parameter, which includes the CMOS capacitance C, β, and the proportionality constant between V_x and f_x.

The clock gating techniques cut off dynamic power consumption during idle periods, by selectively stopping clock signals to portions of the circuit. As a result, during idle periods the dynamic power is negligible and the power consumption is dominated by the static power only [6]. The CG techniques are particularly safe for mission-critical FEs [8], since idle-active transitions introduce very small overhead. For example, the authors in [9] estimated an exit latency of less than 1 µs for the Xeon 5680 processor.

Starting from the above described features of DVFS and CG technologies, the power consumption of an energy-aware FE can be modeled as a function of the forwarding capacity and of the utilization ($\rho \in [0,1]$) as follows:

$$\Phi(\mu_x, \rho) = \Phi_{stat} + K\, \mu_x^{(1+\frac{2}{\gamma})}\, S(\rho) \quad (5)$$

$S(\rho)$ is a function, whose aim is to take into account the scaling effect induced by the alternating active-idle periods in the FE queuing system served by the network processor.

The FE utilization represents the fraction of time spent in the active period. For a queuing system with infinite buffer in equilibrium, it is given by $\rho = \lambda/\mu_x$, where λ is the packet arrival rate of the traffic. If we consider an ideal CG mechanism, the power consumption of the device can be modeled as a linear function with respect to ρ. However, as reported in previous works [2], [10]-[11], due to circuit level inefficiencies, the power consumption of CG-capable devices increases in a concave way according to ρ. Thus, in order to take into account the more accurate modeling of the CG mechanism, we define the function $S(.): \mathbb{R} \rightarrow [0,1]$ as follows:

$$S(\rho) = \rho^{1/\alpha} = (\lambda/\mu_x)^{1/\alpha}, \text{ with } \alpha > 1 \quad (6)$$

As observed in [7] and [12], the processor speed is almost linearly related with the supply voltage; so, in the rest of this paper we will suppose $\gamma = 1$, which implies a cubic relationship between the working speed and the dynamic power consumption. Thus, by substituting (6) in (5), the power consumption of the FE turns out to be:

$$\Phi(\mu_x) = \Phi_{stat} + K\, \mu_x^3\, (\lambda/\mu_x)^{1/\alpha} \quad (7)$$

B. The Optimization Problem

Given the power consumption model described in the previous section, we can formulate an optimization problem.

The objective of the optimization problem is to find the optimal service rate, which minimizes the power-delay product and satisfies the throughput requirements. Note that if we take the average queue traversal delay per packet, the power-delay product represents the average energy consumed by the system to serve a packet.

However, since Φ_{stat} is constant and does not depend on traffic load and processor speed, it makes sense to consider in the product only the energy spent in the dynamic part, which is consumed (in addition to the static component) only when the processor is active (i.e., for a fraction ρ of the time). We consider then the following cost function:

$$J(\mu_x) = \widetilde{\Phi}(\mu_x)\, T(\lambda, \mu_x) \quad (8)$$

where:

$$\widetilde{\Phi}(\mu_x) = K\, \mu_x^3\, (\lambda/\mu_x)^{1/\alpha} \quad (9)$$

and $T(\lambda, \mu_x)$ is defined as:

$$T(\lambda, \mu_x) = \frac{1}{2\mu_x}\, \frac{2-\rho}{1-\rho} \quad (10)$$

Note that Eq. (10) describes the delay function of an M/D/1 queuing system. On one hand, the assumption of Poisson arrival traffic can be a good approximation for highly aggregated traffic [13], as it is in the case of backbone links. On the other hand, despite the possible inaccuracy of the M/D/1 delay approximation, it results to be an effective penalty function with respect to the saturation of the processor capacity.

We solved the optimization problem in the continuous domain by relaxing the discrete variable μ_x. This leads to the following optimization problem:

$$\min_{\mu} \widetilde{\Phi}(\mu)\, \frac{1}{2\mu}\, \frac{2-\rho}{1-\rho} \quad (11)$$

Where $\mu \in \mathbb{R}$.

The optimization problem in (11) admits a unique simple analytical solution:

$$\mu^* = \vartheta \cdot \lambda = \frac{-3+7\alpha+\sqrt{1-10\alpha+17\alpha^2}}{-4+8\alpha}\, \lambda \quad (12)$$

Note that the more efficient is the CG mechanisms ($\alpha \to 1$) the higher the optimal speed, since most of the power can be saved by exploiting idle periods. On the contrary, if $\alpha \to \infty$, (i.e., the CG provides a poor energy gain), the optimal speed decreases to save energy with DVFS. Furthermore, since μ^* is linearly proportional to λ, the energy-aware queue system tends to operate at a constant utilization factor $\rho = \frac{1}{\vartheta}$.

Equation (12) gives the optimal service capacity in the continuous domain, but as explained in Section II-A the hardware implementation of DVFS generally provides a discrete set of couples of V_x and f_x. For this reason, in the next section we provide a practical AR policy, which exploits the result in (12) to adapt the service capacity of the forwarding engine to the load.

III. THE ADAPTIVE RATE POLICY

Starting from the result obtained in the previous section, we consider a simple multi-threshold policy. In detail, given the set of X working frequencies $\{f_0, f_1, \dots, f_X\}$ (where $f_i < f_{i+1}$) and, consequently, the set of associated maximum packet service rates $\{\mu_0, \mu_1, \dots, \mu_X\}$, it is possible to obtain the $X-1$ thresholds of our policy by inverting (12):

$$\lambda_x = \frac{1}{\vartheta}\,\mu_x \tag{13}$$

when the traffic rate λ exceeds the threshold λ_x, the policy increases the speed to μ_{x+1} in order to meet (12). Differently, when the traffic is between λ_{x-1} and λ_x, the policy sets the speed μ_x. In this way the selected speed is always $\mu_x \geq \mu^*$, providing a sub-optimal cost index.

A key element of every AR policy is the traffic prediction mechanism. In general Internet traffic has self-similar and long-range-dependent dynamics and statistical features, which are hard to be accurately captured at any time-scales by traffic models [14]. Sophisticated traffic models also require the collection of traffic data and parameters, generally not available to network devices (e.g., packet inter-arrival time, traffic burstiness, etc.). Furthermore, a complex traffic predictor may require a high computational capacity, which can introduce non-negligible additional power consumption.

For these reasons, we suppose the FE to be only able to count the number of packets, p, received during a time window T_w. The estimated packet rate defined as $\tilde{\lambda} = p/T_w$, is used as an input of the traffic predictor. Similarly to [3] and [15], we decide to use an Exponentially-Weighted Moving Average Predictor (EWMAP). The EWMAP generates the predicted value for the next time interval via a linear combination of the last observed value $O(t-T_w)$ and the previous predicted value $P(t-T_w)$:

$$P(t) = \omega\, P(t-T_w) + (1-\omega)\, O(t-T_w) \tag{14}$$

The weight factor $\omega \in [0,1]$ controls the reactivity of the predictor. In fact, small values of ω result in a more agile predictor. The time window T_w plays an important role for any linear predictor and needs to be carefully selected. A too small T_w may not allow the EWMAP to gather sufficient information to accurately predict the next packet arrival rate. On the other hand, a large time window decreases the reactivity of the predictor to sudden increases of traffic.

The authors in [15] performed and exhaustive performance evaluation of the EWMAP. They tested the predictor by using several real traffic traces of different backbone links, and they conclude that agile predictors are better suited for AR applications, providing an average prediction error of less than 2%.

Although higher values of T_w could produce better performance in terms of prediction error, since the number of available clock frequencies is usually not greater than 16 and service capacity is always set greater than the predicted λ, the capability of quickly detecting the thresholds crossing has an higher priority with respect to the prediction's accuracy. Starting from these results, we set the time window equal to 100 ms and $\omega = 0.2$.

Because of the agility of the used prediction mechanism, our AR policy can produce quite a few frequency transitions, which introduce an additional delay in the packet service time. Usually the time required to change the working frequency in DVFS processors is 10 µs [18] during while no packet forwarding can be executed. In order to moderate the effects of the frequency scaling, we introduced a hysteresis behavior. In detail, after a change in the processor's speed, the policy must waits 1 s (which corresponds to ten times the time window T_w) before reducing again the frequency. In this way, even considering the unlikely case where the traffic continuously crosses up and down thresholds every second for 24 hours, only 0.8640 seconds of processor time are wasted in useless frequency switching.

To evaluate the performance of the proposed policy, we assumed a maximum packet service rate of $\mu_{max} = 1.488$ Mpps, corresponding to the maximum packet arrival rate in a 1 GbE link. Differently, to decide the minimum service rate we have to take into account of the practical implementation limits of DVFS. As reported in [16], DVFS processors limit the minimum voltage to approximately half of the maximum V_x; thus, f_{max}/f_{min} is usually in the range 2-3 [16], [17], [18]. Accordingly, we set the lowest processing speed to $\mu_{min} = 595.3$ kpps. By considering packets of the minimum size (i.e., 64 Byte), μ_{min} corresponds to the 40% of link utilization. Note that $f_{max}/f_{min} = \beta\,\mu_{max}/\beta\,\mu_{min} = 2.5$, which is coherent with the data of several DVFS processors. Regarding the other frequencies, we consider two sets of equally spaced speeds, one composed by 4 clock frequencies and one composed by 13 clock frequencies, denoted as 4-speeds and 13-speeds respectively. Finally, according to the experimental results in reference [10], we set the parameter α=2.

Figure 1 depicts the resulting "autonomic profiles" of our AR policy for both the speed sets compared with the Business As Usual (BAU) case (where no AR mechanism is applied). In detail, Figure 1-(a) shows the FE's packet service rate according to the traffic load[1], while Figure 1-(b) depicts the

[1] In the rest of this paper we will consider the percentage of traffic load calculated with respect to the maximum packet arrival rate in a 1 GbE.

normalized dynamic power calculated with Eq. (7). Note that approximately after the 50%-60% of traffic load, both the versions of the AR policy select the maximum working speed, guarantying the maximum forwarding performance during high traffic periods. In this respect, nothing prevents the designer to increase the parameter ϑ in order to have more conservative autonomic profiles.

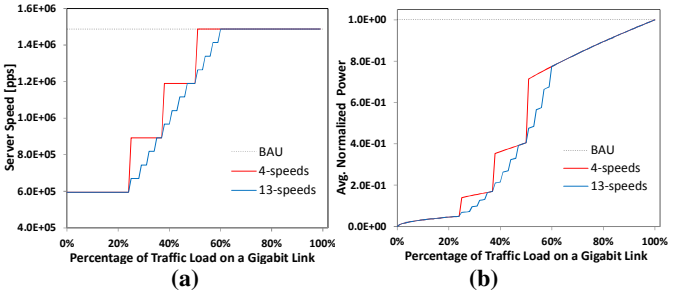


Figure 1. Packet service rate (a) and dynamic power consumption (b) for the following cases: Business As Usual (BAU), adaptive rate with 4 clock frequencies (4-speeds) and adaptive rate with 13 clock frequencies (13-speeds)..

IV. Performance Evaluation

This section provides the performance evaluation of the AR policy described in Section III. In order to evaluate the proposed policy, we developed a discrete-time simulator of a single server queue with DVFS and CG capability.

We assumed that the frequency transitions cause a service interruption of 10 µs. During the service interruption time, packets are backlogged in the queue waiting to be served.

We initially tested our policy by generating Poisson distributed traffic at different average loads. The duration of each simulation run is 10 seconds, and for each run we sampled the packet delay, the queue occupancy and the average dynamic power. We compared the proposed policy with the case where no power saving technologies are applied. The latter case, denoted as BAU, corresponds to set $\mu = \mu_{max}\ \forall\ \lambda$.

Figure 2 shows the average packet delay vs. the average load, the whiskers at each measured point corresponds to the 95th and 5th percentile. Differently from the BAU case, the average delay provided by both the AR policies (i.e., the 4-speeds and 13-speeds versions) shows a non-monotonic behavior for traffic loads $\lambda \leq \sim 60\%$.

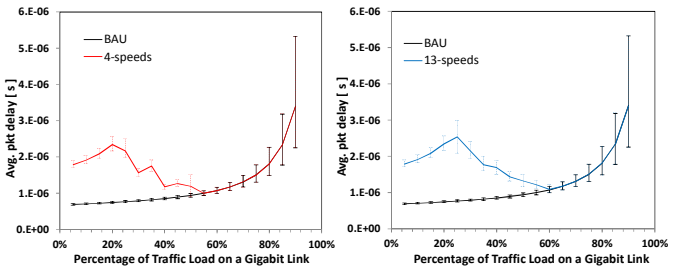


Figure 2. Average packet delay vs. traffic load under the 4-speeds (a) and 13-speeds (b) AR policies compared with the BAU case.

Indeed, according to Figure 1-(a), in low load regime (i.e., below ~25% of link utilization), the policy select the minimum service rate. Therefore, the average system traversal delay starts to growth accordingly to traffic load until the latter exceeds the first threshold λ_0. As a consequence of that, a higher service rate is selected and because of the lower server utilization the average delay decreases.

As one can better observe in Figure 2-(a), this oscillatory behavior of the packet delay is repeated every time the traffic rate exceeds the policy's thresholds. Finally, when the policy sets the maximum service capacity, the packet delay curve returns to increase as in the typical M/D/1/K queue system. The performance provided by both the 4-speeds and 13-speeds policies are very similar, but in the latter case the delay curve is smoother. This is because having a higher number service capacities allows a more fine-grained adaptation to the traffic load levels.

As clarified by the previous figures, in our proposed policy, lower consumption is traded with a small increase of packet delays. However reducing the service capacity can also causes higher queue occupancy. This may lead to buffer overflows and throughput reduction. To address this important issue, we have plotted the average queue's length in Figure 3.

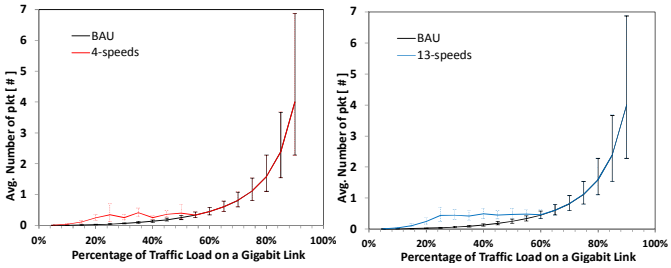


Figure 3. Average number of packets in the queue under the 4-speeds (a) and 13-speeds (b) AR policies compared with the BAU case.

By observing Figure 3 we can note that the queue size remains very close to the BAU case, with very small variations. Furthermore, for traffic loads in the range of 25%-60% the queue occupancy appears to be almost constant with respect to the load. Indeed, the occupancy of a queuing system under Poisson traffic and deterministic service time only depends on ratio λ/μ, but according to our AR policy the packet service time is linearly proportional λ, therefore the queue size tends to be constant, being constant λ/μ.

Regarding the energy efficiency, Figure 4 depicts the normalized dynamic power according to the traffic load.

As expected, the 13-speeds version of the policy provides a smother power profile curve, however, it only provides an additional average power gain of ~4%.

In general, both the two sets of speeds provide good levels of power saving at price of accepting a small loss in the forwarding performance. For instance, by considering an average link utilization of 30% and by only increasing the average packet latency of 1.5 µs, we can potentially save the

~90% of dynamic power (corresponding to the 45% of the overall power for a 65 nm CMOS chip).

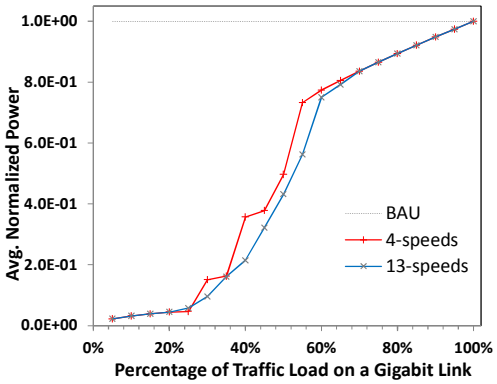


Figure 4. Relative dynamic power consumption according to the traffic load.

The results obtained so far revealed that a good tradeoff between forwarding performance and energy efficiency is experienced when using our optimization policy. However in real scenarios, the AR operates in presence time variant (non-stationary) packet arrival rate. Thus, the only use of stationary traffic is not sufficient to evaluate the system performance.

With this in mind, we perform further simulations by using a 120 second long synthetic traffic trace showed in Figure 5. Specifically, Figure 5 shows the average traffic load according to the simulation time. The traffic trace is divided in 3 second long time slots; in each slot the packet inter-arrival time is exponentially distributed. We tune the traffic profile so that the average link utilization over the duration of the simulation is around 35%. Every 100 ms of simulation, we calculated the average packet latency, the average queue occupancy and the power consumptions.

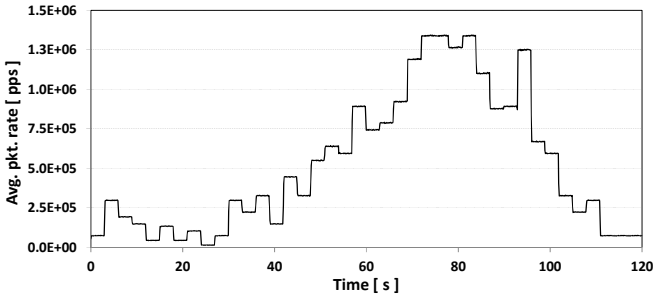
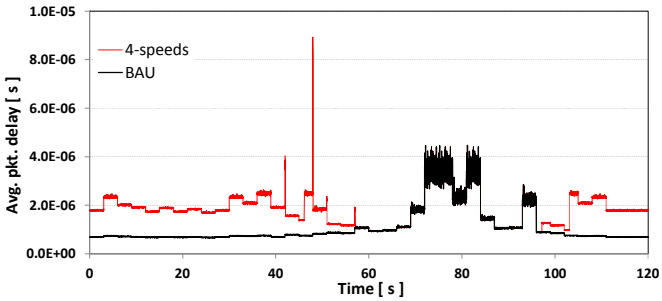


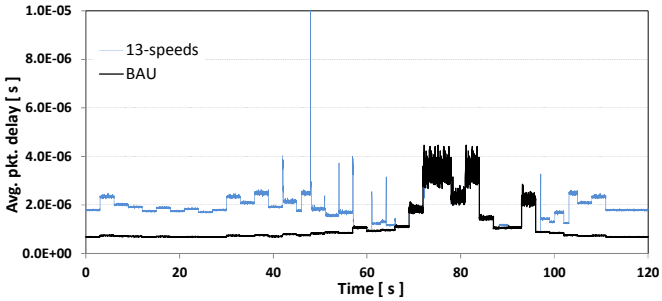
Figure 5. Packet rate of the input traffic trace according to the simulation time.

Figure 6 shows the packet latency during the simulation for both the set of speeds. In both the cases the average packet latency hovers approximately in the 1-10 µs range. However for the 13-speeds case, depicted in Figure 6-(b), we can observe more frequent latency peaks especially in the 40-65 s time window, where the traffic exhibit an high coefficient of variation. In fact, due to the more fine-grained resolution of the 13-speeds set, more variable traffic causes more frequent service interruption periods.

Despite these periods of temporary absence of service, the proposed policy guarantees a very low average number of packets in the queue, (fewer than 6 packets as shown in Figure 7). This prevents buffer overflow and consequently packet loss.

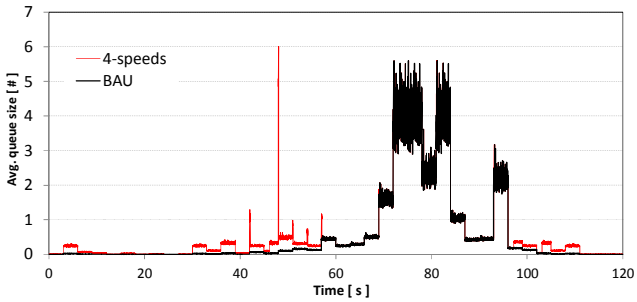


(a)

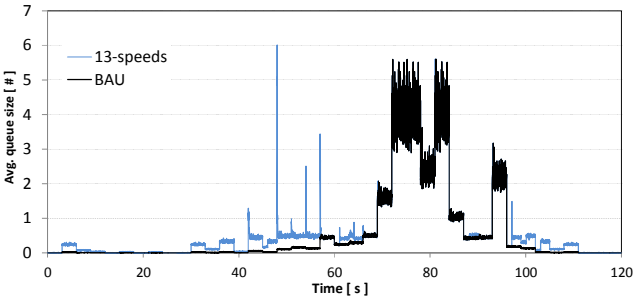


(b)

Figure 6. Average packet delay according to the simulation time under the 4-speeds (a) and 13-speeds (b) AR policies compared with the BAU case.



(a)



(b)

Figure 7. Average number of packets in the queue according to the simulation time under the 4-speeds (a) and 13-speeds (b) AR policies compared with the BAU case.

Finally, Figure 8 shows average normalized power dissipation according to the simulation time. As already noticed in Figure 4, the 13-speeds policy version provides higher savings. This is because the system can works closer to the optimal operating point, thanks to the higher speed resolutions.

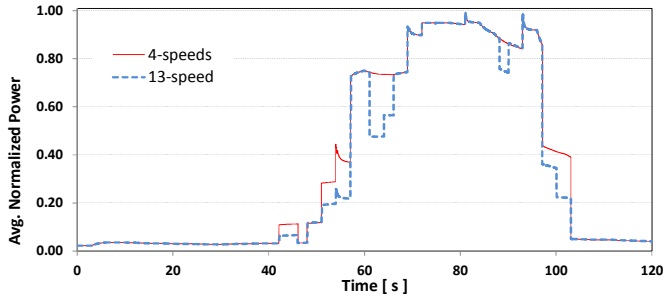


Figure 8. Average dynamic power under the 4-speeds and 13-speeds AR policies according to the simulation time.

However, in our case-study the 4-speeds and 13-speeds policies allow to save the 65% and 67.5% of dynamic power (corresponding to $32.5% and 35% if with consider a static power weight of 50%), respectively. Thus, in the latter case only an additional 2.5% of dynamic power is saved.

Starting from these estimations, it is clear that energy efficiency could be improved through a small number of service capacities. Moreover, if incoming traffic is relatively "unstable", a large set of working frequencies could causes too frequent frequency oscillations, which increase the average packet delay.

V. Conclusion

We considered a forwarding engine whose power consumption can be scaled through DVFS and CG. Our main objective was to improve the power efficiency of the FE without compromising its performance. To this purpose we proposed an optimization framework to dynamically optimize the power-delay product and we designed a lightweight AR policy. The performance evaluation was conducted using a discrete-time simulator. We showed that a significant improve of energy efficiency is experienced when using our policy. The power gain is exchanged with a small increase of packet delays. Findings also show that it is not necessary to support a large number of different clock frequencies, because a number of 4 is enough to achieve significant reductions in the global power consumption. On the contrary, a large number of clock frequencies could lead to more frequent frequency oscillations, which can compromise the forwarding performance.

Acknowledgment

This work has been supported by the the FIRB project Futuro e Ricerca Greening the Network (GreenNet).

References

[1] Global e-Sustainibility Initiative (GeSI), "SMART 2020: Enabling the Low Carbon Economy in the Information Age", http://www.theclimategroup.org/assets/resources/publications/Smart2020Report.pdf.

[2] R. Bolla, R. Bruschi, A. Carrega, F. Davoli, D. Suino, C. Vassilakis, A. Zafeiropoulos, "Cutting the Energy Bills of Internet Service Providers and Telecoms through Power Management: An Impact Analysis," *Computer Networks,* vol. 56, no. 10, pp. 2320-2342, July 2012.

[3] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, D. Wetherall, "Reducing Network Energy Consumption via Sleeping and Rate-Adaptation," *Proc. 5th USENIX Symp. on Networked Systems Design and Implementation* (NSDI'08), USENIX Association, Berkeley, CA, USA, pp. 323-336.

[4] R.S. Tucker, R. Parthiban, J. Baliga, K. Hinton, R.W. Aire, W.V. Sorin, "Evolution of WDM Optical IP Networks: A Cost and Energy Perspective," *IEEE J. Lightw. Technol*., vol. 27, no. 3, pp. 243-252, Feb. 2009.

[5] D. Nguyen, A. Davare, M. Orshansky, D. Chinnery, B. Thompson, K. Keutzer, "Minimization of Dynamic and Static Power through Joint Assignment of Threshold Voltages and Sizing Optimization [Logic IC Design]," *Proceedings 2003 International Symp. on Low Power Electronics and Design,. ISLPED '03*, pp.158-163, Aug. 2003.

[6] S. Henzler, *Power Management of Digital Circuits in Deep Sub-Micron CMOS Technologies*, Springer Series in Adv. Microel., 2007.

[7] K. Li, "Scheduling Parallel Tasks on Multiprocessor Computers with Efficient Power Management," *Proc. IEEE International Symp. on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW),* pp.1-8, April 2010.

[8] Y. Luo, J. Yu, J. Yang L. N. Bhuyan, "Conserving Network Processor Power Consumption by Exploiting Traffic Variability," *ACM Trans. Archit. Code Optim.*, vol. 4, no. 1, March 2007.

[9] L. Niccolini, G. Iannaccone, S. Ratnasamy, J. Chandrashekar, L. Rizzo,. "Building a Power-Proportional Software Router," *Proc. 2012 USENIX Annual Technical Conference (USENIX ATC'12)*, USENIX Association, Berkeley, CA, USA, 2007.

[10] C. H. Lien, Y. W. Bai, M. B. Lin, "Estimation by Software for the Power Consumption of Streaming-Media Servers," *IEEE Trans. Instrumentation and Measurement,* vol. 56, no. 5, pp. 1859-1870, Oct. 2007.

[11] C. J. Tang, M. R. Dai, "Dynamic Computing Resource Adjustment for Enhancing Energy Efficiency of Cloud Service Data Centers," *Proc. IEEE/SICE International Symp. on System Integration (SII 2011),* pp.1159-1164, Dec. 2011.

[12] D. Zhu, R. Melhem, B. R. Childers, "Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multiprocessor Real-Time Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 14, no. 7, July 2003.

[13] T. Karagiannis, M. Molle, M. Faloutsos, and A. Broido, "A nonstationary poissonview of internet traffic," in Proc. of 23rd Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2004., vol. 3, pp. 1558-1569, March 2004.

[14] T. Karagiannis, M. Molle, and M. Faloutsos, "Long-range dependence ten years of internet traffic modeling," IEEE Internet Computing, vol. 8, pp. 57-64, Sept 2004.

[15] M. Mandviwalla, N. F. Tzeng, "Energy-Efficient Scheme for Multiprocessor-Based Router Linecards," *Proc. International Symp. on Applications and the Internet, SAINT 2006,* pp.-163, 23-Jan. 2006.

[16] B. Zhai, D. Blaauw, D. Sylvester, K. Flautner, "Theoretical and Practical Limits of Dynamic Voltage Scaling," *Proc. 41st ACM Annual Design Automation Conf. (DAC '04)*, New York, NY, USA, pp. 868-873.

[17] Intel Xeon Processor 3400 Series, LGA1156 Socket: Thermal/Mechanical Design Guide.
http://www.intel.com/content/www/xr/en/processors/xeon/xeon-3400-lga1156-socket-thermal-design-guide.html

[18] Enhanced Intel® SpeedStep® Technology for the Intel® Pentium® M Processor.
http://download.intel.com/design/network/papers/30117401.pdf.

[19] MAWI Woring Group Traffic Archive, Sample Point F, available at http://mawi.nezu.wide.ad.jp/mawi/samplepoint-F/20080318/.