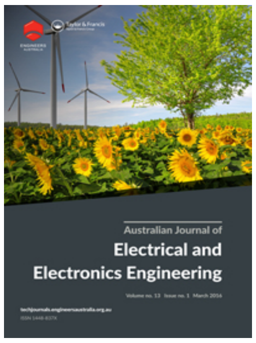


Australian Journal of Electrical and Electronics Engineering

ISSN: 1448-837X (Print) 2205-362X (Online) Journal homepage: http://www.tandfonline.com/loi/tele20

Network interface power management and TCP congestion control: a troubled marriage

Carla Panarello, Alfio Lombardo, Giovanni Schembra, Michela Meo, Marco Mellia & Marco Ajmone Marsan

To cite this article: Carla Panarello, Alfio Lombardo, Giovanni Schembra, Michela Meo, Marco Mellia & Marco Ajmone Marsan (2016) Network interface power management and TCP congestion control: a troubled marriage, Australian Journal of Electrical and Electronics Engineering, 13:1, 67-76, DOI: 10.1080/1448837X.2015.1093678

To link to this article: http://dx.doi.org/10.1080/1448837X.2015.1093678

Published online: 10 Jan 2016.

AUSTRALIAN JOURNAL OF ELECTRICAL AND ELECTRONICS ENGINEERING, 2016
VOL. 13, NO. 1, 67–76
http://dx.doi.org/10.1080/1448837X.2015.1093678

Network interface power management and TCP congestion control: a troubled marriage

Carla Panarello[a], Alfio Lombardo[b], Giovanni Schembra[b], Michela Meo[c], Marco Mellia[c], Marco Ajmone Marsan[c,d]

[a]CNIT – Research Unit of University of Catania, Catania, Italy; [b]DIEEI – University of Catania, Catania, Italy; [c]DET – Politecnico di Torino, Torino, Italy; [d]Institute IMDEA Networks, Madrid, Spain

ABSTRACT
Optimizing the trade-off between power saving and Quality of Service in the current Internet is a challenging research objective, whose difficulty stems also from the dominant presence of Transmission Control Protocol (TCP) traffic, and its elastic nature. More specifically, recent works support the possibility of improving energy efficiency of network devices by modulating switching and transmission capacity according to traffic load, whereas TCP traffic is in turn adaptive to the available resources. In a previous work, we have shown that an intertwining exists between capacity scaling approaches and TCP congestion control. In this paper, we investigate the reasons of such intertwining, and we evaluate how and how much the dynamics of the two algorithms affect each other's performance. More specifically, we will show that such an interaction is essentially due to the relative speed of the two algorithms, which determines the conditions for the successful or unsuccessful coexistence of the two mechanisms.

ARTICLE HISTORY
Received 15 April 2014
Accepted 05 February 2015

KEYWORDS
Energy saving; power management; capacity scaling; TCP; quality of service

1. Introduction

In today's Internet, a significant fraction of the energy consumed by network devices is wasted, because no or very little proportionality exists between energy consumption and device utilization; in other words, the energy consumption of network devices is today largely independent of the carried traffic. For this reason, recent research works advocate the possibility of improving energy efficiency of network devices by modulating their switching and transmission capacity according to their current traffic load (Nedevschi et al. 2008; Bolla et al. 2009; 2011; Wierman et al. 2009).

Of course, in the designer intention, the reduction of the overall network energy consumption achieved by modulating switching and transmission capacity should in no way adversely affect network performance. This quite obvious constraint is not trivially met in the case of TCP traffic. Indeed, in the case of Transmission Control Protocol (TCP), congestion control algorithms and energy-saving mechanisms may interact, with the effect of decreasing both Quality of Service (QoS) and energy savings. More specifically, while on the one side, the TCP congestion control algorithm adapts the TCP source sending rate to the available network resources, on the other, green routers activate their power management schemes by scaling their service rate, that is, their switching and transmission capacity, according to traffic. In turn, the variation of a green router service rate determined by power management schemes induces changes in the network available resources, which affect TCP congestion control, and so on, with a loop whose effects are difficult to predict. Since most of the traffic in today's Internet is carried by TCP, the investigation of the behaviour of the resulting closed loop and of the effect of its time constants is of fundamental importance for energy-efficient networking research, and is the objective of this work.

In previous works Panarello (2013,2012), we presented a preliminary exploration of the interplay between the congestion control algorithm of TCP and capacity scaling approaches. Simulation results for a simple scenario showed that mutual reactions exist, and impact performance, in terms of both energy saving and QoS. In this paper, we conduct a much more detailed analysis, to understand how the dynamics of both mechanisms affect each other's performance. The results collected in this paper indicate that the interaction is essentially due to the overlap of the two closed-loop controls, with different time constants, that is, the relative speed of the two algorithms plays a fundamental role, determining the conditions for the successful or unsuccessful coexistence of the two mechanisms.

In particular, in Section 2, we present the problem and discuss related work. Section 3 presents the power consumption model for capacity scaling approaches. In Section 4, we describe the case study we use to

CONTACT Carla Panarello cpana@dieei.unict.it
The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n. 257740 (Network of Excellence "TREND").

investigate on the intertwining between capacity scaling and TCP congestion control. The simulation set-up is described in Section 5. Numerical results are presented in Section 6 and, finally, in Section 7, we draw our conclusions.

2. Problem statement and background

Most of the schemes proposed in the literature to reduce network energy consumption have been tested using real-world network topologies and traffic workloads. However, up to now, to the best of our knowledge, traffic workloads were reproduced by artificially generating traffic according to the measured patterns or statistics (see e.g. Gunaratne et al. 2008; Nedevschi et al. 2008; Christensen et al. 2010), but ignoring the feedback that energy-saving schemes may have on the traffic itself. In other words, these works do not take into account the possibility that changes in the instantaneous offered traffic rate may be caused by the effect that energy-saving mechanisms have on rate controlled sources, such as TCP. In fact, on the one hand, energy-saving mechanisms modify the amount of resources available at network devices according to traffic, but, on the other hand, TCP traffic sources adapt their sending rate according to the available resources. Therefore, unexplored looped reactions occur when the TCP congestion control mechanism is coupled with power management capabilities. The goal of this paper is to investigate about the successful or unsuccessful coexistence of power management schemes, and in particular capacity scaling approaches, and TCP congestion control.

The issue of TCP energy efficiency has attracted the interest of researchers since the late 90's. The early works study the energy consumption of TCP connections over wireless channels and compare the performance of different TCP versions using different approaches (see e.g. Zorzi and Rao 1999; Tsaoussidis et al. 2000). More recent works, such as Bolla et al. (2013), Reviriego, Sanchez-Macian and Maestro (2011), investigate on the impact of TCP on network energy consumption, yet not considering the feedback produced by power management schemes on TCP. Other works, such as Sassu, Scarso and Cuomo 2012, assess whether the reduction of the overall network energy consumption can be achieved without adversely affecting network performance, and are thus more in line with our approach. However, they do not consider the impact of speed scaling. Two recent papers that look at speed scaling and TCP are Nedevschi et al. (2008); Gunaratne et al. (2008); however, they do not address the issue of the interaction between the speed scaling algorithm and the TCP congestion control algorithm, like we do in this paper. This issue was, to the best of our knowledge, only addressed in our previous paper Panarello (2012).

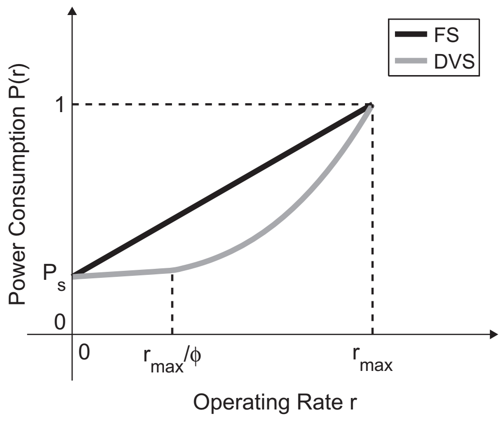


Figure 1. Power Model.

3. Power consumption model for capacity scaling approaches

Studies conducted in the last few years demonstrate that part of the energy consumed in networks is wasted, due to the fact that, on the one hand, the energy consumption of network devices remains practically constant regardless their utilization, whereas on the other hand, networks are provisioned for worst-case or busy-hour load, and their average utilization remains low, with long idle-times present (Jardosh 2007).

For these reasons, recent works advocate the possibility of reducing power consumption of network devices by modulating switching and transmission capacity according to their traffic load. The idea is to provide the network interface with a number of performance states corresponding to different link rates, and to adopt some algorithm able to determine the actual traffic rate and choose among the available link rates, the one closer to the traffic rate. The idea relies on the consideration that devices operating more slowly show a lower power consumption: e.g. ethernet links at 1 and 10Gb/s dissipate about 2 and 20 W, respectively (Gunaratne et al. 2008). Moreover, realizing the frequency scaling (FS) on a network interface, allows to exploit the possibility of regulating the operating voltage accordingly, and scaling the power consumption cubically with the operating frequency. This technique is known in the literature as dynamic voltage scaling (DVS) (Zhai et al. 2004).

The adoption of the above techniques allows to express the power consumption of a device as a function of the link rate r. However, as typical for actual devices, a portion of the power consumption is assumed to be constant and independent of the operating rate. Therefore, the power $P(r)$ consumed by a network device can be computed as: $P(r) = P_s + f(r)$, where P_s is the static amount of power and $f(r)$ represents the rate-dependent portion of consumption (Nedevschi et al. 2008). For FS and DVS, $f(r)$ is a linear and cubic function of r, i.e. $f(r) = O(r)$ and $f(r) = O(r^3)$, respectively. For DVS, an additional constraint needs to be considered because, in practice, there is a minimum rate below which scaling the link rate offers no further voltage reduction. Therefore, a maximum scaling factor ϕ is introduced, such that $f(r) = O(r^3)$ for $r \in [r_{MAX}/\phi; r_{MAX}]$. Figure 1 shows the qualitative trend of power consumption, $P(r)$, in devices supporting either FS or DVS.

Let us note that the numerical evaluation of the power saving, according to the above power consumption model, is hardware and technology dependent and a deep analysis on it is out of the scope of this paper. Nevertheless, modelling the power consumption as a monotonically increasing function of the link rate, allows us to give an indicative measure of the energy saving introduced by a capacity scaling mechanism, through the evaluation of the reduction of the average link rate with respect to the maximum link capacity.

4. Case study

In this section, we describe the selected case study. In order to understand how TCP congestion control and capacity scaling interact, and affect each other's performance, we consider a simple bottleneck network topology, like the one depicted in Figure 2. Note that the goal of this work is to identify and understand the reasons of that interaction, thus it is worth to start this study from a very simple topology, which leaves aside incidental and third-party effects on the dynamics of the two mechanisms under study. This condition is fundamental to relate causes and effects, without adding noisy and secondary phenomena produced by a complex network topology.

We look at two scenarios. In the first one, named TCP+PC, we have both power control in node $N1$ and TCP traffic with its congestion control. In the second scenario, named TCPonly, node $N1$ does not implement any power control, and TCP implements the only traffic control mechanism. Let us note that we address this latter scenario as a reference for our study.

To better understand which dynamics come into play, we initially focus our attention on a single TCP connection (Section 6.1).

In order to also analyse the interplay between TCP congestion control and capacity scaling with a more realistic setting for TCP traffic, in Section 6.3, we consider the fact that TCP traffic may not fully utilize the resources of node $N1$ because of the presence of bottlenecks elsewhere in the network. To model such a context, we modify the bandwidth available to the TCP flow along its path, so as to introduce higher variations on its sending rate. In the network topology of Figure 2, this corresponds to selecting different values for the capacity of the link between nodes $N2$ and $N3$.

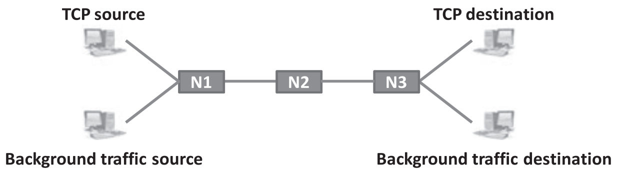


Figure 2. Network topology.

Note that the TCP congestion control provides a quick reduction of the TCP sending rate in the case of losses or congestion detection, but the increase in the TCP sending rate is always gradual. However, in our study it can be interesting to also look at the case of a rapid increase in the arrival rate at router $N1$, to see how it may affect the performance of both TCP and capacity scaling. For this reason, we introduce in Section 6.2, a bursty background traffic which follows the same path as our reference TCP connection (see Figure 2).

In all our simulations, for the scaling of switching and transmission capacity within node $N1$, we consider the power management mechanism proposed in Nedevschi et al. (2008), and there referred as practRA. The goal of the practRA algorithm is to determine the service rate closest to the arrival rate, among a set of available values. To this purpose, this technique predicts the future arrival rate at time t[1], $\hat{r}_f$, by using an exponentially weighted moving average of the measured history of past arrivals. Moreover, both the current buffer length q and the current service rate r_i of the green router are used to estimate the potential queuing delay in the case the service rate $\hat{r}_f$ is used, so as to avoid violating a given delay constraint, d. Due to the arrangement procedures occurring when the service rate of the green router is requested to change, the device cannot send packets for δ seconds after each transition. So, in order to avoid the costs of a high number of rate transitions, these can occur only if at least a minimum time interval, greater than δ, passed since the previous transition. In the following, we will refer to this time interval as *minimum Rate Change Interval mRCI*.

In formulae, the algorithm works as follows:

- A link operating at rate r_i with current queue size q increases its rate to r_{i+1} if and only if $\left(\frac{q}{r_i} > d\right)$ or $\left(\frac{\delta\hat{r}_f+q}{r_{i+1}} > d - \delta\right)$: these conditions assure (i) that, keeping the service rate r_i, the time needed to drain the queue of length q would not be greater than the delay constraint d, or in other words, that the delay constraint d would not be violated keeping the current service rate r_i, (ii) to allow enough time for a link rate increase, so avoiding late attempts of increasing the link rate after violation of the delay constraint d.
- A link operating at rate r_i with current queue size q decreases its rate to r_{i-1} if and only if $(q = 0)$ and $(\hat{r}_f < r_{i-1})$: these conditions assure that (i) the link rate is increased only if the queue is empty, and (ii) oscillations between rates are avoided, that is, that switching to a lower rate does not immediately

lead to larger queues which, in turn, would cause immediate rate increase.

Note that a detailed analysis of the power-saving capability of the above power management algorithm is out of the scope of this paper, also because it is hardware and technology dependent. Nevertheless, an indicative measure of the energy saved by means of a capacity scaling mechanism can be provided, as described in Section 3, by evaluating the reduction of the average service rate with respect to the maximum output link capacity, provided that the power consumption profile of a green router is supposed to be a monotonically increasing function of its service rate (see Figure 1).

5. Simulation set-up

In this section, we describe the simulation set-up. The duration of each simulation run is 300 seconds. The MTU length is 1500 bytes. The transport protocol is TCP NewReno. TCP connections are long lived, and transmit maximum size packets as allowed by the window size. The buffer size of routers $N1$, $N2$ and $N3$ is set equal to the bandwidth-delay product.

The round trip propagation delay varies in each simulation run in the set RTT $= \{1, 10, 50, 100\}$ ms. The maximum capacity of the link between nodes $N1$ and $N2$ is set to 10 Mb/s, while the service rate of $N1$ is set by the capacity scaling algorithm with a granularity of 1 Mb/s in the range $[1, 10]$ Mb/s. The interval between two consecutive service rate updates (minimum Rate Change Interval, $mRCI$) varies in the set $mRCI = \{10, 50, 100, 500, 1000\}$ ms.

The capacity of the link between nodes $N2$ and $N3$ is initially set to 10 Mb/s. However, in order to consider how the presence of bottlenecks in the network impacts the interaction between TCP and capacity scaling, we have also considered, in Section 6.3, the case where the available bandwidth along the link between nodes $N2$ and $N3$ varies during the simulation time. More specifically, the available bandwidth in such a link is uniformly distributed in the range $[1, 10]$ Mb/s, and the duration of the interval between changes of available bandwidth is exponentially distributed with average $BN = \{5 \cdot 10^{-3}, 5 \cdot 10^{-2}, 5 \cdot 10^{-1}, 5\}$ s.

Moreover, as mentioned in Section 4, in order to consider also the case where rapid increases in the arrival rate at node $N1$ may affect the performance of both TCP and capacity scaling, we have introduced, in Section 6.2, a bursty background traffic whose burst duration is distributed exponentially with average $bON = \{1, 10, 50, 100\}$ ms, and cycles every second.

The simulation parameters are summarized in Table 1.

Next, we present results in terms of both QoS and energy saving. As far as QoS is concerned, we consider as performance parameter the reduction of the TCP throughput in the TCP+PC case with respect to the TCPonly case, and the percentage of packet loss in both cases. The performance in terms of energy saving is hardware and technology dependent. However, an idea of the effectiveness of the capacity scaling algorithm in reducing the energy consumption is given through the evaluation of the reduction of the service rate with respect to the maximum link capacity: higher rate reductions translate into larger energy savings.

Moreover, an important parameter to evaluate the efficiency of the capacity scaling mechanism is the number of changes of service rate: every change of rate implies a cost (again dependent from the hardware technology), so that the efficiency of the mechanism is reduced as the number of change increases.

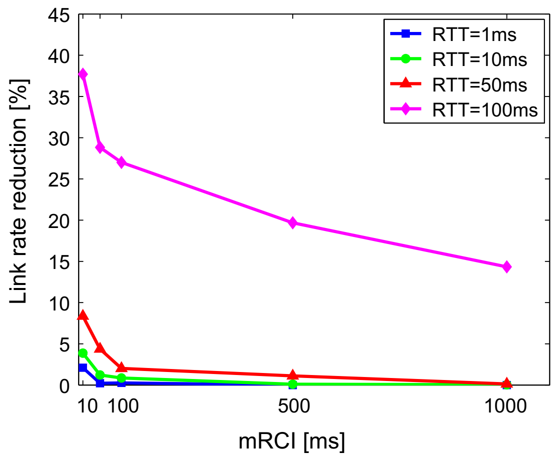
6. Numerical results

In this section, we show how the intertwining between the TCP congestion control algorithm and the capacity scaling mechanism affects each other's performance. To this purpose, we conducted extensive simulations with the ns-2.30 simulator (The network simulator - ns2). In particular, in Section 6.1, we analyse results in the case of a single persistent TCP source, without background traffic and without bottlenecks elsewhere in the network. The impact of a background traffic is evaluated in Section 6.2. Finally, in Section 6.3, we present results in the case a bottleneck is present in the network.

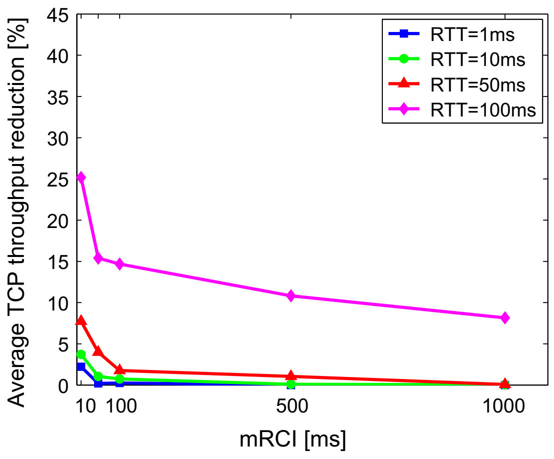
6.1. Single TCP source

In this section, we discuss results in the case of a single persistent TCP source, without background traffic and without bottlenecks elsewhere in the network. Since both TCP and capacity scaling are feedback-based mechanisms, with temporal parameters RTT and $mRCI$, respectively, we demonstrate that the relative values of these two parameters determine how the two algorithms interact and affect each other's performance.

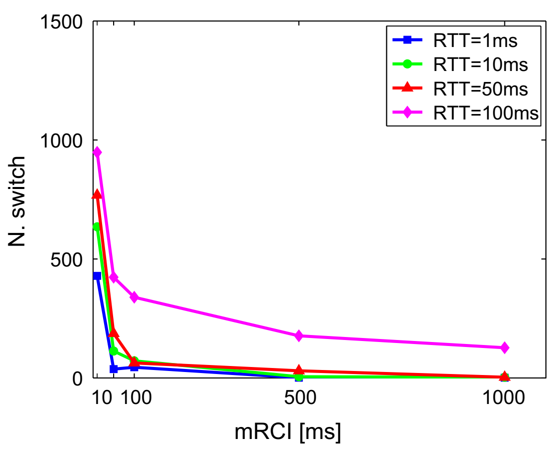
In Figure 3, we present the average service rate chosen by the capacity scaling mechanism (Figure 3(a)), the reduction of the TCP throughput in the TCP+PC case with respect to the TCPonly case (Figure 3(b)), the number of service rate changes made by the algorithm (Figure 3(c)), and the percentage of packet loss in both the TCP+PC and TCPonly cases (Figure 3(d)). Results show that for small values of RTT, and for large values of the $mRCI$ parameter, the average service rate chosen by the capacity scaling algorithm is very close to the maximum link capacity. In these cases, no significant energy saving is present, and at the same time no QoS degradation is introduced by the capacity scaling mechanism. Indeed, the reduction of the TCP throughput with respect to the TCPonly case is almost zero, and the percentage of packet loss is almost the same as in the TCPonly case. Note, however, that when small values of the $mRCI$ parameter are considered, a small service



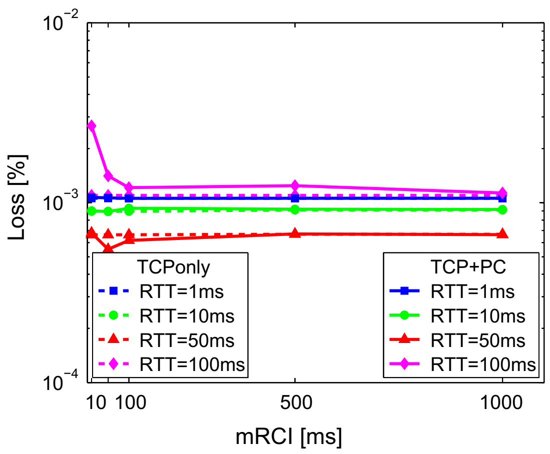
(a) Average Link Rate chosen by the capacity scaling algorithm



(b) Average TCP throughput reduction in the *TCP + PC* case with respect to the *TCPonly* case

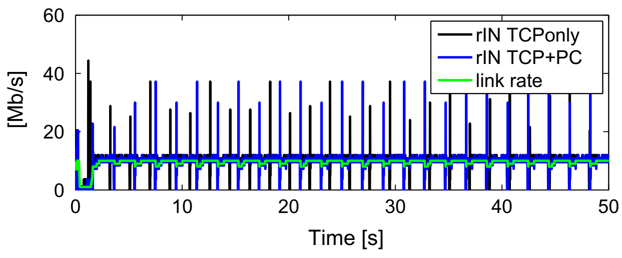


(c) Number of Link Rate changes

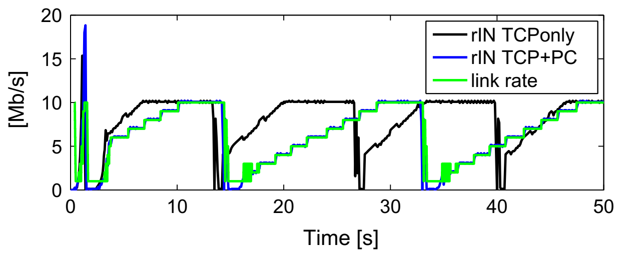


(d) Loss

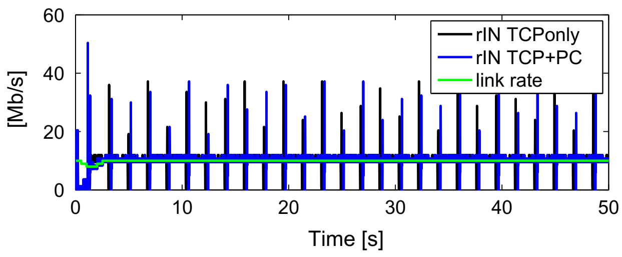
Figure 3. Single TCP source.



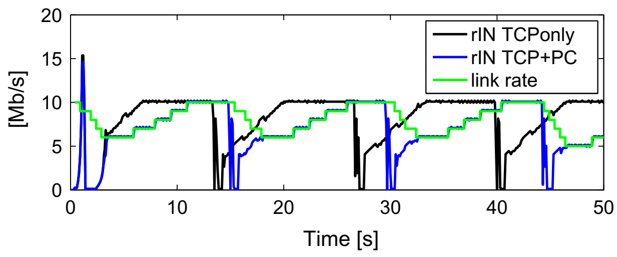
(a) $RTT = 10\text{ms}$ - $mRCI = 10\text{ms}$



(b) $RTT = 100\text{ms}$ - $mRCI = 10\text{ms}$



(c) $RTT = 10\text{ms}$ - $mRCI = 500\text{ms}$



(d) $RTT = 100\text{ms}$ - $mRCI = 500\text{ms}$

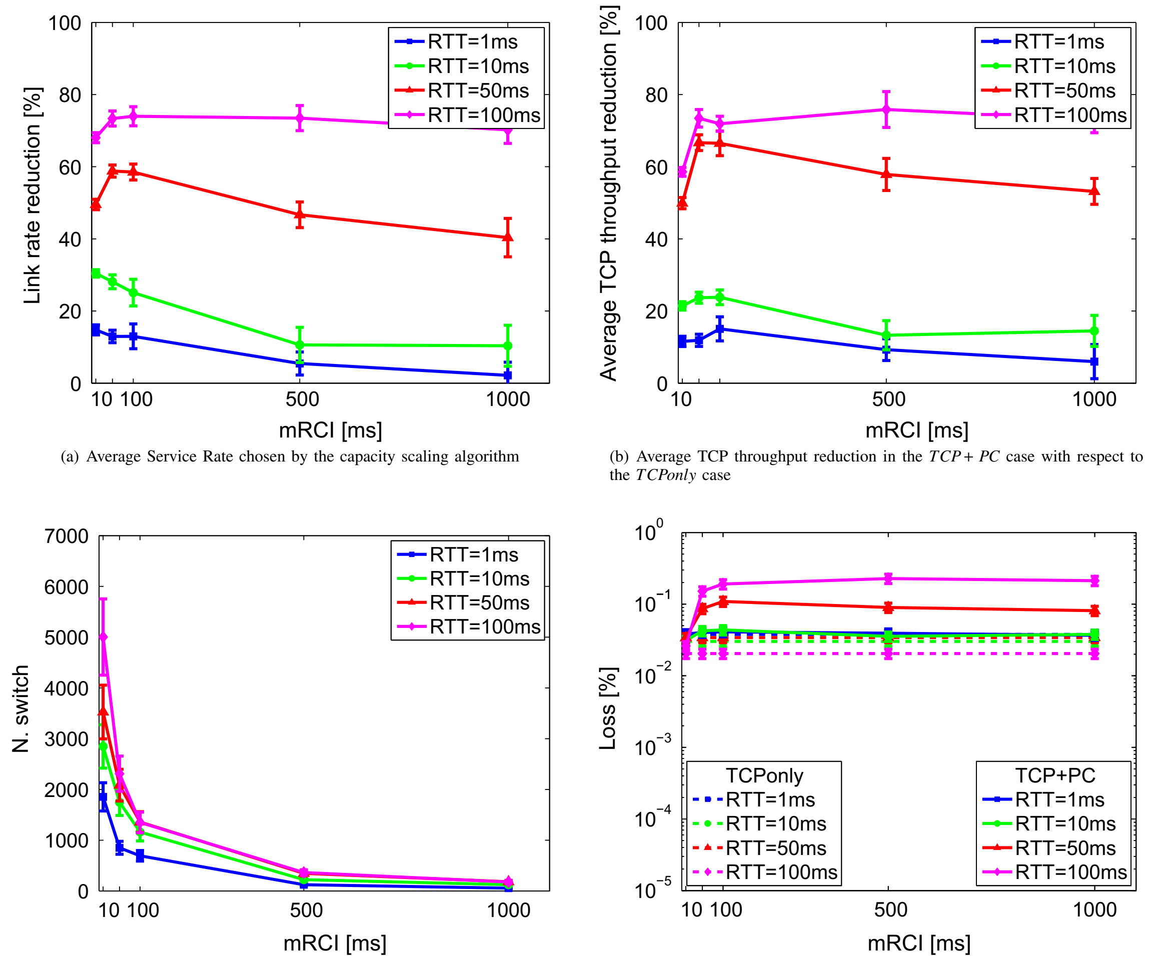
Figure 4. Single TCP source – comparison between the temporal evolution of the service rate chosen by the capacity scaling algorithm and the temporal evolution of the arrival rate at node $N1$ in the TCP+PC and TCPonly cases.

rate reduction and a slight performance degradation is present.

Instead, when higher values of RTT are considered (e.g. RTT = 100 ms, magenta lines in Figure 3), the

Table 1. Simulation parameters.

Simulation Time	300s
Maximum Link Capacity (*N*1-*N*2)	10Mb/s
Maximum Link Capacity (*N*2-*N*3)	10Mb/s
Maximum Transfer Unit (*MTU*)	1500B
Round-Trip Time (RTT)	$\{1, 10, 50, 100\}ms$
minimum Rate Change Interval (*mRCI*)	$\{10, 50, 100, 500, 1000\}$ms
Background traffic	
Burst duration distribution	Exponential
Average burst duration (*bON*)	$\{1, 10, 50, 100\}$ms
Background Traffic Cycle	1s
Bottleneck link N2-N3	
Available bandwidth distribution	Uniform
Available bandwidth range	$[1, 10]$Mb/s
Time interval distribution between bandwidth changes	Exponential
Average time interval (*BN*) between bandwidth changes	$\{5 \cdot 10^{-3}, 5 \cdot 10^{-2}, 5 \cdot 10^{-1}, 5\}$s



(a) Average Service Rate chosen by the capacity scaling algorithm

(b) Average TCP throughput reduction in the *TCP+ PC* case with respect to the *TCPonly* case

(c) Number of Service Rate changes

(d) Loss

Figure 5. Background traffic (bON=100 ms).

service rate chosen by the capacity scaling algorithm is about 20–40% lower than the maximum link capacity (Figure 3(a)), meaning that some energy saving is present. However, the number of service rate changes is large (Figure 3(c)), meaning that the efficiency of the mechanism decreases as the RTT increases. Additionally, the reduction of the throughput of the TCP source, in the TCP+PC case with respect to the TCPonly case, is higher than in the previous cases (Figure 3(b)), and because of this the loss probability becomes higher (Figure 3(d)), so the action of the capacity scaling negatively affects the TCP performance. This performance degradation is higher for small values of the *mRCI* parameter.

In order to understand the reasons of such results, Figure 4 presents a comparison between the temporal evolution of the service rate chosen by the capacity scaling algorithm (green line) and the temporal evolution

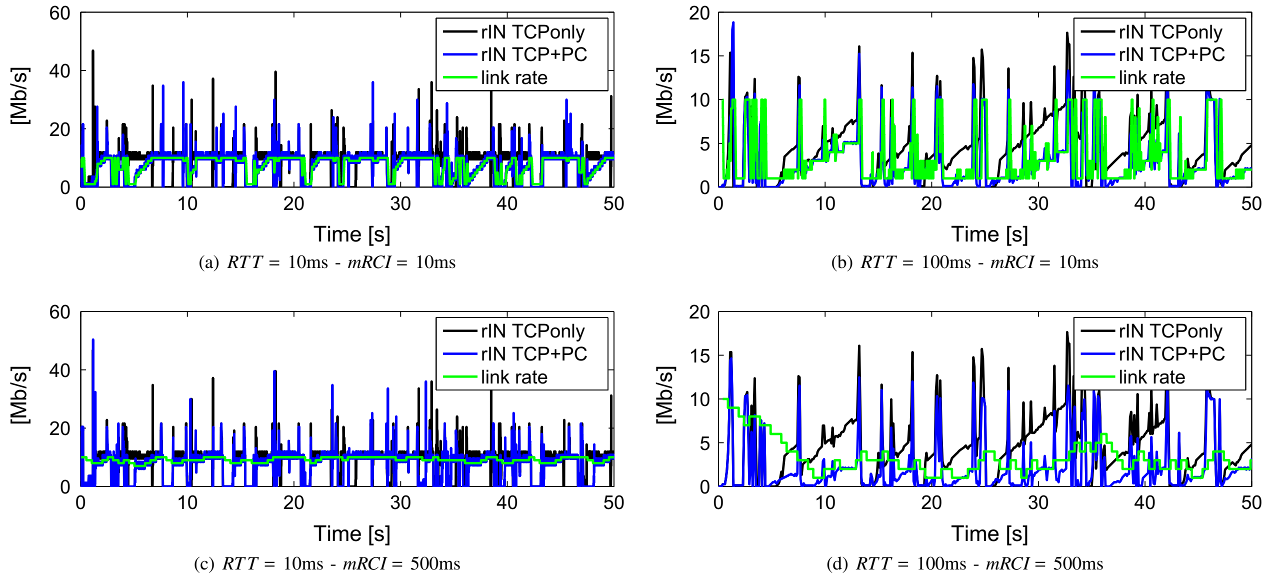


Figure 6. Background traffic (bON=100 ms): comparison between the temporal evolution of the service rate chosen by the capacity scaling algorithm and the temporal evolution of the arrival rate at node $N1$ in the TCP+PC and TCPonly cases.

of the arrival rate in the TCP+PC and TCPonly cases (blue and black lines, respectively). Note that, in the case of single TCP source, the arrival rate at node $N1$ corresponds to the TCP sending rate[2].

Let us start by considering the case where RTT is small. Figure 4(a) and c show that the TCP source achieves an average sending rate almost equal to the maximum available bandwidth (10 Mb/s), and the service rate provided by the capacity scaling mechanism follows the TCP sending rate and stabilizes on that value. Note that, when the *mRCI* parameter presents small values (Figure 4(a)), the service rate exhibits small variations around the average value. Such variations disappear as the *mRCI* parameter becomes higher (Figure 4(c)). In fact, for high values of the *mRCI* parameters, the capacity scaling algorithm is less reactive: it works on an average of the arrival rate calculated on longer time intervals and does not need to modify the service rate, so the higher the *mRCI* parameter, the more similar the behaviour to a non-capacity scaling mechanism.

On the contrary, when RTT is large (Figures 4(b) and (d)), the consequent higher bandwidth-delay product makes the TCP source sending rate widely variable. Therefore, the TCP source is not able to completely exploit the available bandwidth of 10 Mb/s, so the average TCP sending rate is lower, and the capacity scaling algorithm can reduce the service rate, thus providing some energy saving. However, as we noted before, in these cases, the reduction of the service rate by the capacity scaling mechanism heavily affects the TCP performance. This performance degradation is higher for small values of the *mRCI* parameter. Indeed, while for the TCP congestion control, high values of RTT result in wide and slow variations of the TCP sending rate, the capacity scaling with small *mRCI* parameters quickly modifies the service rate according to the TCP sending rate variation. So, every time the TCP congestion control tries to increase the TCP sending rate, it is hampered by the service rate previously chosen by the capacity scaling mechanisms according to the previous (lower) TCP sending rate. This is evident in Figure 4(b), where it is possible to see that, after a TCP sending rate reduction (e.g. at the time instant 14, due probably to some loss), the TCP sending rate takes about 6 s to reach again the 10Mb/s value in TCPonly case, whereas it takes about 14 s in TCP+PC case.

Let us now investigate the effects of larger values of the *mRCI* parameter, in the case of large RTT. As mentioned before, the performance degradation is lower when *mRCI* increases. However, note that the choice of a higher value of *mRCI* does not represent actually a better choice, as demonstrated in Figure 4(d). Indeed, when the TCP sending rate decreases as a consequence of some loss detection, the service rate does not decrease accordingly. Therefore, the service rate remains higher than the arrival rate, and a waste of energy occurs. However, in the meanwhile, the TCP sending rate can increase freely. Unfortunately, as soon as the arrival rate becomes equal to the service rate, the further increase in the sending rate by the TCP source is hampered again by the service rate. We can see that the rising slope of the blue and green lines, related to the TCP sending rate in the TCP+PC case, and the service rate, respectively, are almost the same as in the case of a small *mRCI* value (Figure 4(b)). Therefore, we can conclude that the improvement of performance for increasing values of *mRCI*, is actually due only to the slower decrease in the service rate. This is actually a very particular case, where the variation of the arrival rate at node $N1$ is quite regular due to the presence of a single TCP source, which does not compete for the available bandwidth with other traffic flows and does not suffer for the presence of bottlenecks elsewhere in the network. We will see in the following, that the presence of disturbing elements, like background traffic or bottlenecks, reverse these results, with performance becoming worst when *mRCI* increases.

6.2. Background traffic

In this section, we evaluate the impact of bursty background traffic on the interaction between TCP and capacity scaling mechanisms. We generated results for different dynamics of the background traffic. Confidence intervals at 95% confidence level were evaluated for all cases. For the sake of space, here we present only the case of $bON = 100$ ms. However, similar results are obtained in the other cases that we studied.

Figure 5(a) shows that the service rate is substantially lower than the maximum link capacity of 10 Mb/s, specially for larger values of RTT. This behaviour may assure some energy saving; however, the cost in terms of number of service rate changes increases slightly as RTT increases, but substantially as *mRCI* increases (Figure 5(c)). Moreover, the loss increases with respect to the TCPonly case (Figure 5(d)) and the reduction of the TCP throughput is huge, specially for large values of RTT (Figure 5(b)).

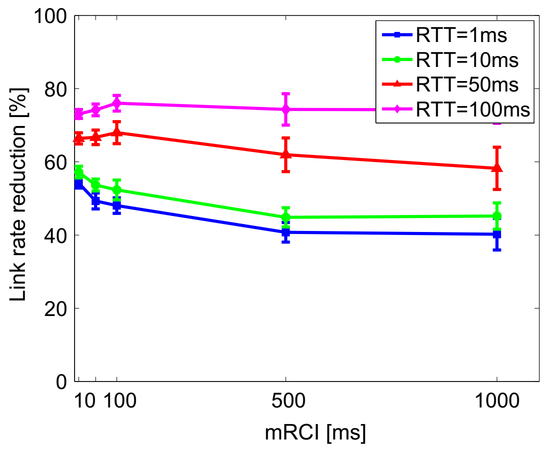
The reason for such results is that in the presence of bursty background traffic, the variation of the arrival rate at node $N1$ becomes greater than for the case with only a single TCP source (compare Figures 6 and 4). However, the capacity scaling mechanism may not notice in due time the increase in the arrival rate due to a spike of background traffic and the service rate may remain for long time lower than the arrival rate (Figure 6), forcing, in the meanwhile, losses and throughput reduction. The performance is worse when large values of RTT are considered (Figures 6(b) and 6(d)). Indeed, the larger the RTT, the longer the time needed by TCP to discover the new bandwidth availability. At the same time, large values of *mRCI* (Figures 6(c) and 6(d)) introduce additional delay for the actions of the capacity scaling mechanism, that follow the TCP variations too late.

Note however, that in the case of small values of *mRCI*, the performance is anyhow bad: the reduction of the TCP throughput is high, as well as the packet loss, specially for large values of RTT (Figures 5(b) and 5(d)). These results are due to the fact that the capacity scaling mechanism tries to follow “almost instantaneously” the arrival rate variations due to the variations of the TCP sending rate (Figures 6(a) and 6(b)), even before TCP is able to discover the new available bandwidth, whose value will be altered by the (low) service rate forced by the capacity scaling algorithm.

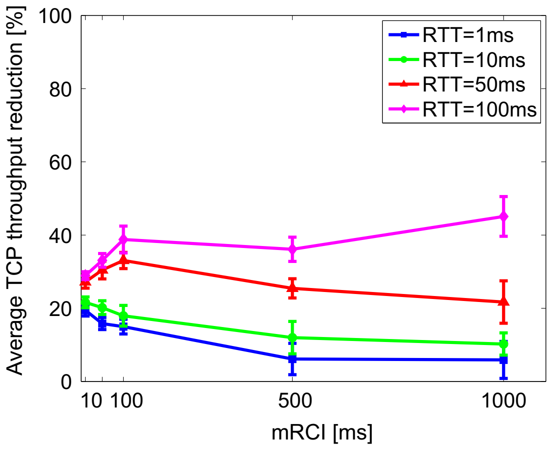
6.3. Bottlenecks elsewhere

Finally, we analysed the case in which a bottleneck is present elsewhere in the network, both in the presence of a single TCP source, and in conjunction with background traffic. We have considered several dynamics for the variation of the available bandwidth in the bottleneck link (from node $N2$ to node $N3$). However, for the sake of space, again we only present results for just one representative case. More specifically, the results in Figure 7 refer to the case where a background traffic is present ($bON = 100$ ms) and the bottleneck available bandwidth varies with temporal parameter $BN = 50$ ms.

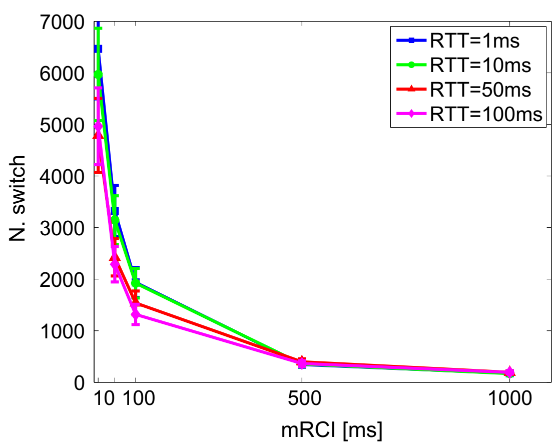
When a bottleneck is present, results are better than for the case with background traffic considered in Section 6.2. Indeed, the energy saving is higher, specially for small values of RTT, and the TCP throughput reduction is lower, specially for large values of RTT (compare Figures 5 with 7). This is due to the fact that the presence of a bottleneck in the network reduces the average available bandwidth (remember that it varies uniformly between 1 and 10 Mb/s), thus forcing the reduction of the TCP sending rate and, therefore, a lower average arrival rate at node $N1$, with respect to the case where no bottleneck is present. This is evident when comparing Figures 8 and 6. Let us focus, for example, on Figures 8(b) and 6(b): when a bottleneck is present (Figure 8(a)), after some loss detection, and the consequent reduction of the TCP sending rate, the arrival rate at node $N1$ rises to no more than 5 Mb/s (apart from the spike of background traffic); instead, when no bottleneck is present (Figure 6(a)), the arrival rate increases to about 10 Mb/s. Therefore, in this specific condition of bottleneck in the network, the error margin between the average arrival rate at node $N1$ and the service rate chosen by the capacity scaling algorithm is generally lower, thus producing better performance with respect to the case shown in Section 6.2. Even so, the analysis of the temporal evolution of the service rate and of the arrival rate in both the TCPonly and TCP+PC cases, reveals the same issues found in Section 6.2. Indeed, again, if the capacity scaling algorithm is too slow (i.e. for large values of *mRCI*), the need for reducing or increasing the service rate may be noticed too late by the capacity scaling mechanism, and in the meanwhile a waste of energy or a performance degradation may be produced (see e.g. Figure 8, time instants 8 and 35, respectively). On the contrary, if the capacity scaling mechanism is too fast (i.e. for low values of *mRCI*), it follows almost instantaneously the TCP sending rate variations. Remember that when TCP experiments losses, it reduces suddenly the sending rate, and then gradually increases it again. So, if the capacity scaling reduces the service rate suddenly as well, TCP ends up to be limited either by the bottleneck bandwidth, or by the (too early) reduced service



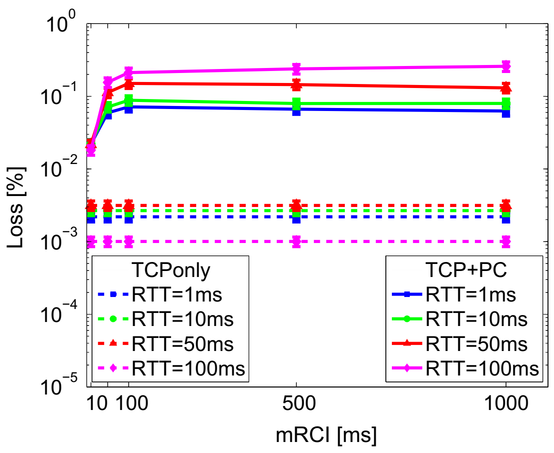
(a) Average Service Rate chosen by the capacity scaling algorithm



(b) Average TCP throughput reduction in the *TCP + PC* case with respect to the *TCPonly* case

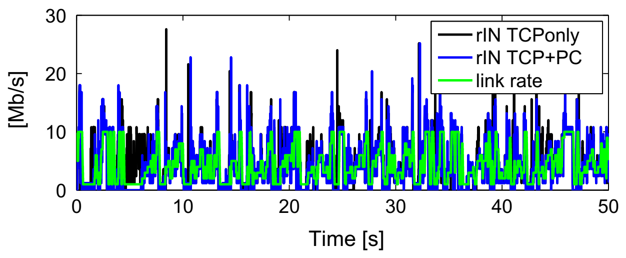


(c) Number of Service Rate changes

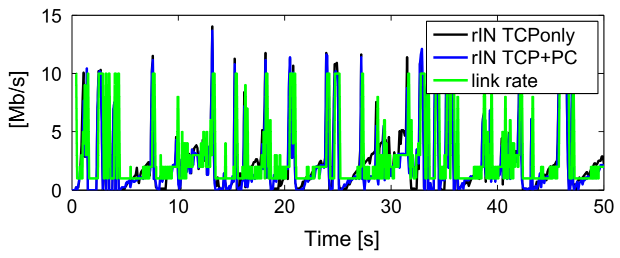


(d) Loss

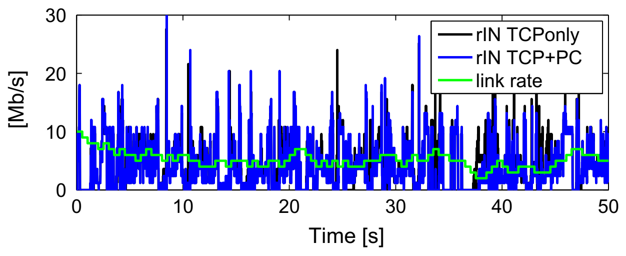
Figure 7. Bottleneck (BN=50 ms)–Background Traffic (bON=100 ms).



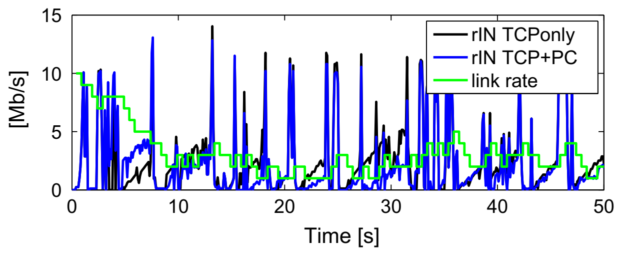
(a) *RTT* = 10ms - *mRCI* = 10ms



(b) *RTT* = 100ms - *mRCI* = 10ms



(c) *RTT* = 10ms - *mRCI* = 500ms



(d) *RTT* = 100ms - *mRCI* = 500ms

Figure 8. Bottleneck (BN=50 ms)–Background Traffic (bON=100 ms): Comparison between the temporal evolution of the service rate chosen by the capacity scaling algorithm and the temporal evolution of the arrival rate at node $N1$ in the TCP+PC and TCPonly cases.

rate of the capacity scaling mechanism (see e.g. Figure 8(b), around time instant 30). As a consequence, even achieving some energy saving (Figure 7(a)) at a cost, in terms of number of service rate changes (Figure 7(c)), similar to the case where no additional bottlenecks are considered, the performance degrades: the percentage of lost packets increases (Figure 7(d)) and the TCP throughput is reduced (Figure 7(b)).

7. Conclusions

In this paper, we have looked at the interaction between the congestion control algorithm of TCP and the capacity scaling algorithms proposed for the improvement of the energy efficiency of Internet nodes.

Our simple simulation set-ups show that in a number of cases, the two types of algorithms may interact in quite a negative fashion, with a drastic performance degradation of TCP flows. This is essentially due to the overlap of the two closed-loop controls, with different time constants. More specifically, results collected in this paper show that, most of the time, for increasing values of RTT and decreasing values of *mRCI*, both energy saving and TCP performance degradation increase.

More simulation experiments are necessary to further characterize the phenomenon that we have discussed in this paper, considering the simultaneous presence of a number of TCP connections with different values of RTT, as well as more complex network topologies, with the final objective of devising a TCP-friendly capacity scaling algorithm for the next generations of green Internet nodes.

Notes

1. To simplify notation, we omit the explicit indication of dependence on time t.
2. The spikes in the figures are related to the increase in the TCP sending rate before a loss detection and the consequent sending rate reduction

References

Bolla, R., R. Bruschi, F. Davoli, and A. Ranieri. 2009. "Performance Constrained Power Consumption Optimization in Distributed Network Equipment." *Green Communications Workshop*, Dresden, Germany, June.

Bolla, R., R. Bruschi, F. Davoli, and F. Cucchietti. 2011. "Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Netweork Infrastuctures." *IEEE Communications Surveys & Tutorials* 13 (2): 223–244.

Bolla, R., R. Bruschi, O. M. Jaramillo Ortiz, and P. Lago. 2013. *The Energy Consumption of TCP*, 3rd ACM/IEEE Internat. Conf. on Future Energy Systems (e-Energy 2013)2013. Berkeley, CA, USA, May.

Christensen, K., P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. A. Maestro. 2010. "IEEE 802.3az: the road to energy efficient ethernet." *IEEE Communications Magazine* 48 (11): 50–56.

Gunaratne, C., K. Christensen, and B. Nordman. 2005. "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed." *International Journal of Network Management* 15 (5): 297–310.

Gunaratne, C., K. Christensen, B. Nordman, and S. Suen. 2008. "Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR)." *IEEE Transactions on Computers* 57 (4): 448–461.

Jardosh, A. P., G. Iannaccone, K. Papagiannaki, and B. Vinnakota. 2007. "Towards an Energy-Star WLAN Infrastructure." In *Eighth IEEE Workshop on Mobile Computing Systems and Applications, HotMobile 2007*, 85–90.

Nedevschi, S., L. Popa, G. Iannaccone, S. Ratnasamy, D. Wetherall. 2008. "Reducing Network Energy Consumption via Sleeping and Rate-Adaptation." In *USENIX/ACM NSDI 08*, San Francisco, USA, April .

Panarello, C., M. Ajmone Marsan, A. Lombardo, M. Mellia, M. Meo, and G. Schembra. 2012. "On the intertwining between capacity scaling and TCP congestion control." In *Third International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy) 2012*, Madrid, May 9-11, 1–4.

Panarello, C., A. Lombardo, G. Schembra, M. MeoM. Mellia, and M. Ajmone Marsan. 2013. "Power Management and TCP Congestion Control: Friends or Foes." In *Proc. of SSEEGN2013*, Christchurch, New Zealand, November 20–22.

Reviriego, P., A. Sanchez-Macian, J. A. Maestro. 2011. "On the Impact of the TCP Acknowledgement Frequency on Energy Efficient Ethernet Performance, NETWORKING." Workshops Lecture Notes in Computer Science 6827 (2011): 265–272.

Sassu, A., C. Scarso, and F. Cuomo. 2012. *TCP behavior over a greened network, Sustainable Internet and ICT for Sustainability (SustainIT 2012)*, Pisa, October 4–5, 1–5.

The network simulator - ns2. :http://www.isi.edu/nsnam/ns/.

Tsaoussidis, V., H. Badr, X. Ge, and K. Pentikousis. 2000. "Energy/throughput Tradeoffs of TCP Error Control Strategies." *Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, Antibes, 106–112.

Wierman, A., L. H. Andrew, and A. Tang. 2009. "Power-Aware Speed Scaling in Processor Sharing Systems." *IEEE INFOCOM 2009*, Rio de Janeiro, Brazil, April.

Zhai, B., D. Blaauw, D. Sylvester, and K. Flautner. 2004. *Theoretical and Practical Limits of Dynamic Voltage Scaling, DAC 2004*. San Diego, CA, USA, June .

Zorzi, M., and R. R. Rao. 1999. In *Is TCP energy efficient?, 1999 IEEE International Workshop on Mobile Multimedia Communications (MoMuC '99)*, San Diego, CA, USA, 198–201.