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

Network Interface Power Management and TCP Congestion Control: a Troubled Marriage

Carla Panarello*, Alfio Lombardo†, Giovanni Schembra†, Michela Meo‡, Marco Mellia‡, Marco Ajmone Marsan‡§
*CNIT - Research Unit of University of Catania, Italy - cpana@dieei.unict.it
†DIEEI - University of Catania, Italy - {lombardo,schembra}@dieei.unict.it
‡DET - Politecnico di Torino, Italy - {meo,mellia,ajmone}@tlc.polito.it
§Institute IMDEA Networks - Madrid, Spain

***Abstract*—Optimizing the tradeoff between power saving and Quality of Service (QoS) in the current Internet is a challenging research objective, whose difficulty stems also from the dominant presence of TCP traffic, and its elastic nature. 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 overlap of the two closed loop controls, with different time constants.**

I. 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 [1], [2], [3], [4]. In such case, the power consumption of a network device can be expressed as a function of the current traffic rate r. Normally, we assume that the power $P(r)$ consumed by a network device carrying traffic at rate r can be expressed as: $P(r) = P_s + f(r)$, where P_s is the static amount of power necessary to power on the device at zero traffic load, and $f(r)$ represents the rate-dependent portion of power consumption.

Of course, in the designer intention, the reduction of the overall network energy consumption achieved by modulating switching and transmission capacity should be achieved without adversely affecting network performance. This quite obvious request is not trivially achieved in the case of TCP traffic. Indeed, in the case of TCP, congestion control algorithms and energy-saving mechanisms may interact, with the effect of decreasing both 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 behavior 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 a previous work [5], 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 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 II, we describe the case study we use to investigate on the intertwining between capacity scaling and TCP congestion control. The simulation setup is described in Section III. Numerical results are presented in Section IV. In Section V we discuss related work, and finally, in Section VI we draw our conclusions.

II. 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 Fig. 1. 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.

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").

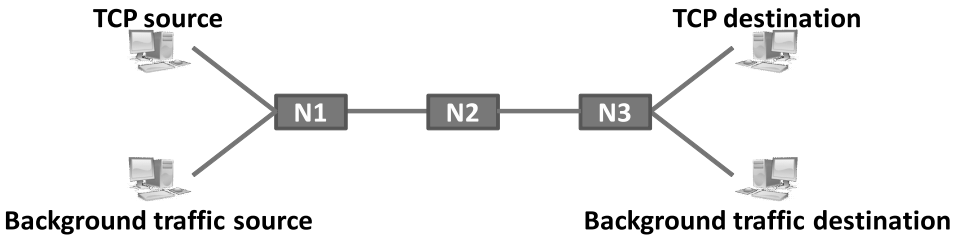


Fig. 1. Network topology

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

In order to also analyze the interplay between TCP congestion control and capacity scaling with a more realistic setting for the TCP traffic, in Section IV-C, 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 Fig. 1 this corresponds to selecting different values for the capacity of the link between nodes $N2$ and $N3$.

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 of 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 of 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 IV-B, a bursty background traffic which follows the same path as our reference TCP connection (see Fig. 1).

In all our simulations, for the scaling of switching and transmission capacity within node $N1$ we consider the power management mechanism proposed in [4], 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 (EWMA) 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*.

[1]To simplify notation, we omit the explicit indication of dependence on time t.

TABLE I
SIMULATION PARAMETERS

Simulation Time	300s
Maximum Link Capacity ($N1$-$N2$)	10Mb/s
Maximum Link Capacity ($N2$-$N3$)	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	$\{5\cdot10^{-3},5\cdot10^{-2},5\cdot10^{-1},5\}$s

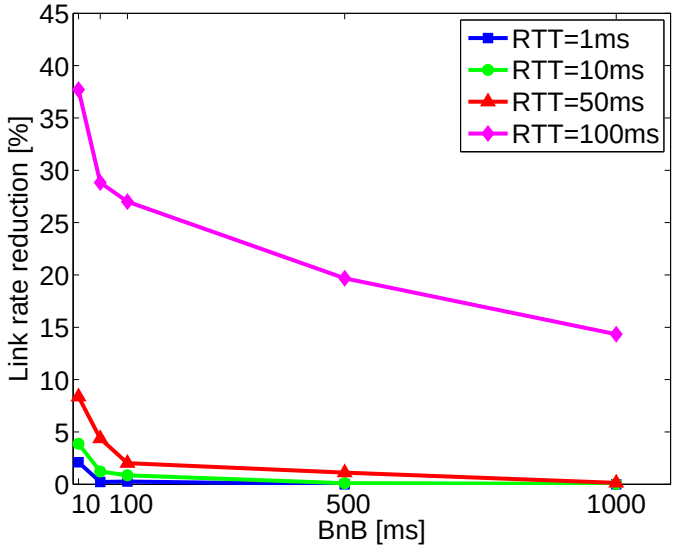
In formulae, the algorithm works as follows:

- A link operating at rate r_i with current queue size q increases its rate to r_{i+1} iff $\left(\frac{q}{r_i} > d \text{ OR } \frac{\delta\hat{r}_f+q}{r_{i+1}} > d-\delta\right)$;
- A link operating at rate r_i with current queue size q decreases its rate to r_{i-1} iff $(q=0 \text{ AND } \hat{r}_f < r_{i-1})$.

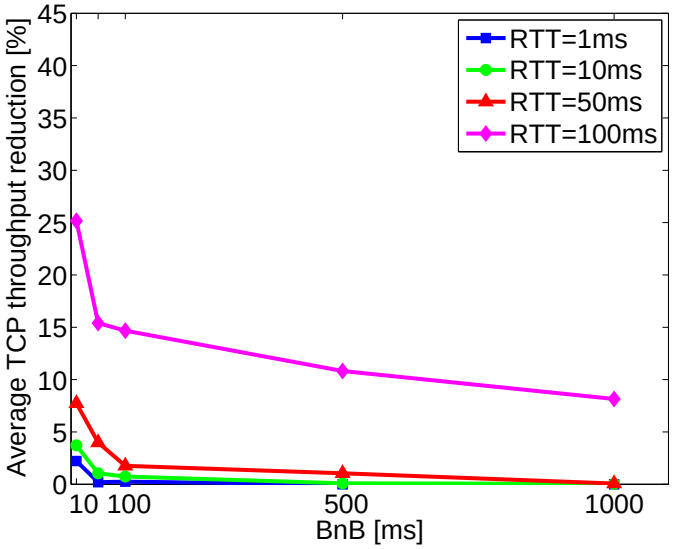
Note that a detailed analysis of the power saving capability of the above power management algorithm is out of the scope of this paper, also because it is hardware and technology dependent. Nevertheless, an indicative measure of the energy saved by means of a capacity scaling mechanism can be provided 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. This is the case for the most relevant hardware technologies available today, such as Frequency Scaling (FS) and Dynamic Voltage Scaling (DVS) [7]. In these cases, the power $P(r)$ consumed by a network device can be computed as $P(r) = P_s + f(r)$, where the rate-dependent portion of the power consumption $f(r)$ is a linear or cubic function of r, i.e. $f(r) = O(r)$ and $f(r) = O(r^3)$, for FS and DVS respectively, whereas P_s is the static amount of power [4].

III. SIMULATION SETUP

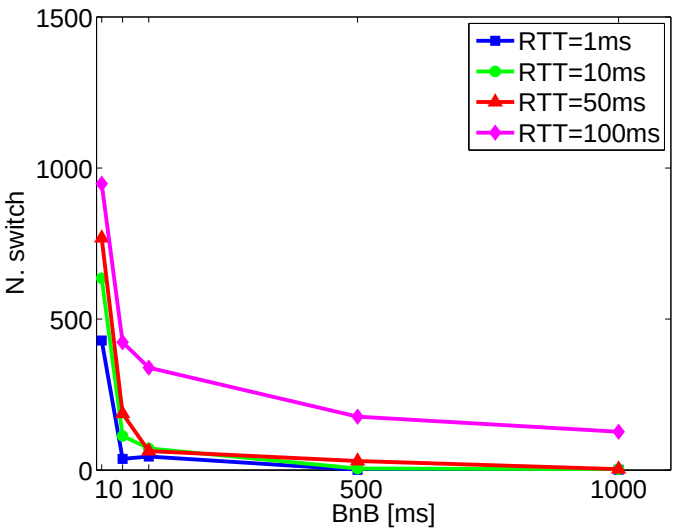
In this section we describe the simulation setup. The duration of each simulation run is 300 seconds. The MTU length is 1500 bytes. The transport protocol is TCP NewReno. TCP connections are long-lived, and transmit maximum size



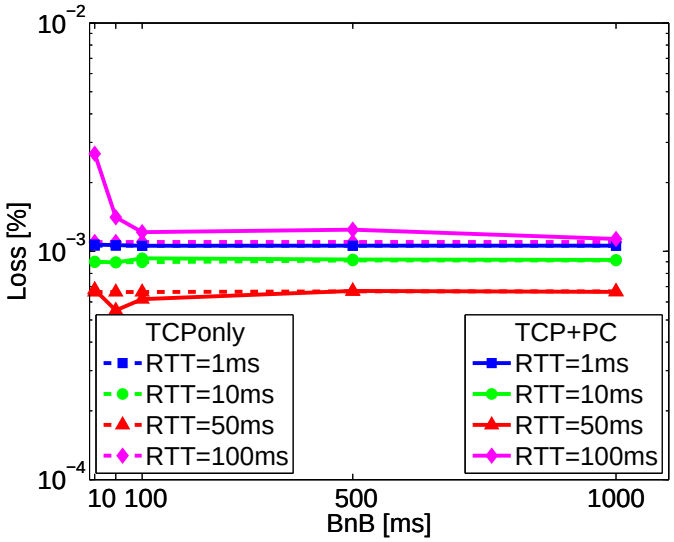
(a) Average Link Rate chosen by the capacity scaling algorithm



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



(c) Number of Link Rate changes



(d) Loss

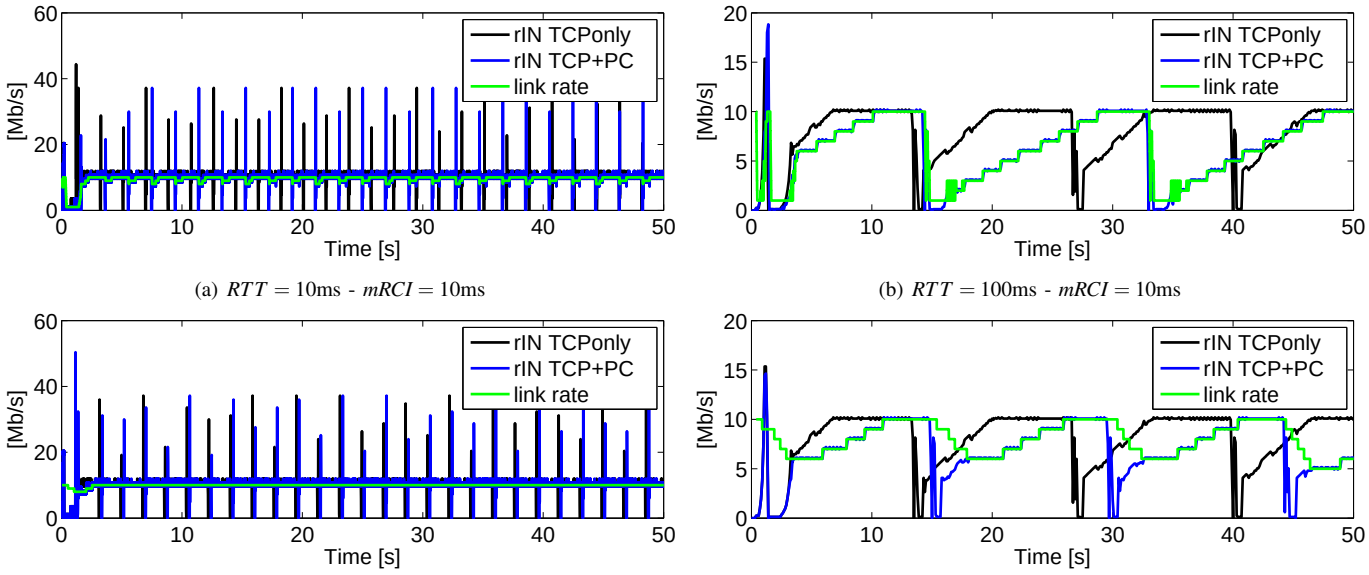
Fig. 2. Single TCP source

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



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

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

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

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

Fig. 3. 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

The simulation parameters are summarized in Table I.

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.

IV. Numerical results

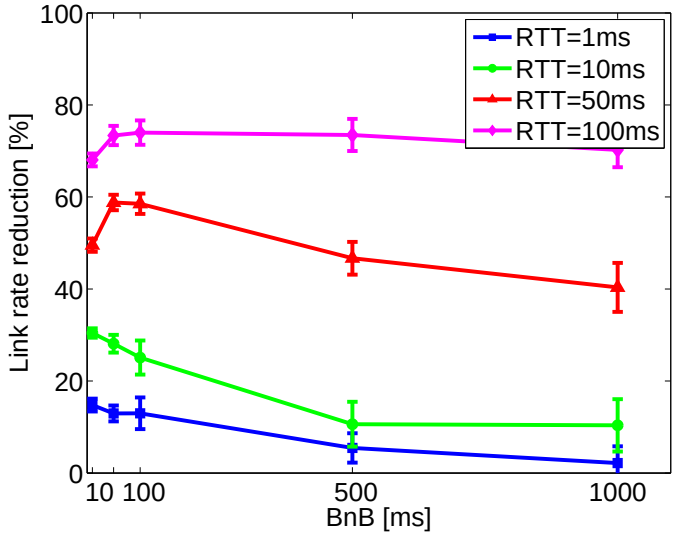
In this section we show how the intertwining between the TCP congestion control algorithm and the capacity scaling mechanism affects each other's performance. To this purpose we conducted extensive simulations with the ns-2.30 simulator [6]. In particular, in Section IV-A we analyze results in the case 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 IV-B. Finally, in Section IV-C, we presents results in the case a bottleneck is present in the network.

A. Single TCP source

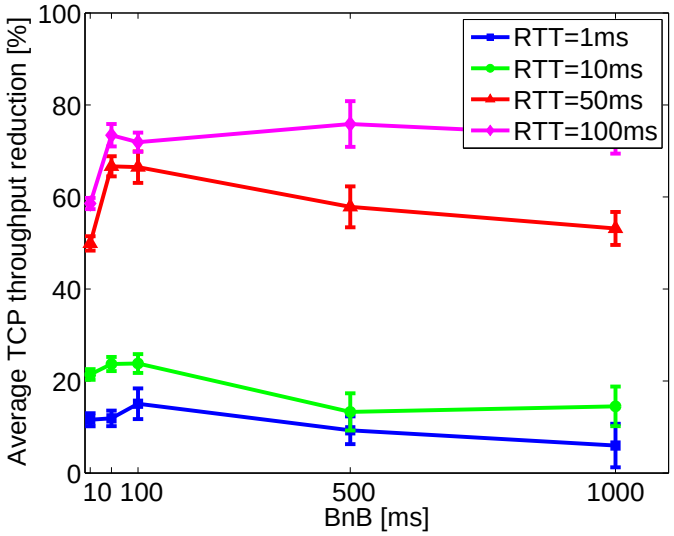
In this section we discuss results in the case of a single persistent TCP source, without background traffic and without bottlenecks elsewhere in the network. Since both TCP and capacity scaling are feedback-based mechanisms, with temporal parameters RTT and $mRCI$ respectively, we demonstrate that the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

In Fig. 2 we present the average service rate chosen by the capacity scaling mechanism (Fig. 2(a)), the reduction of the TCP throughput in the $TCP+PC$ case with respect to the $TCPonly$ case (Fig. 2(b)), the number of service rate changes made by the algorithm (Fig. 2(c)), and the percentage of packet loss in both the $TCP+PC$ and $TCPonly$ cases (Fig. 2(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 rate reduction and a slight performance degradation is present.

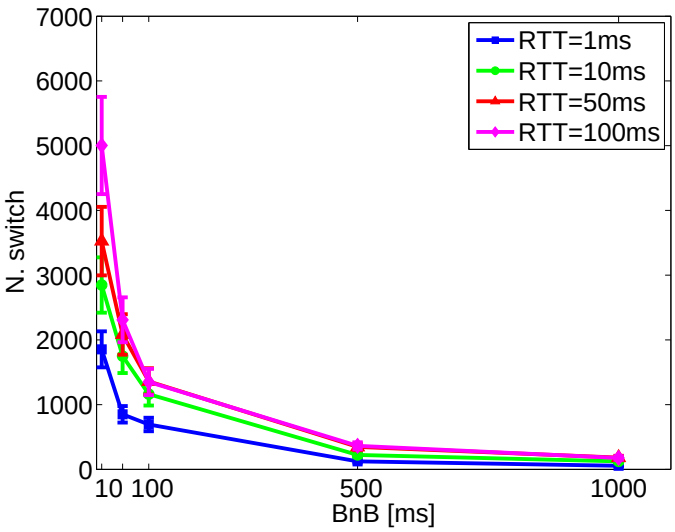
Instead, when higher values of RTT are considered (e.g. $RTT = 100$ ms, magenta line in Fig. 2(a) and Fig. 2(b)), the service rate chosen by the capacity scaling algorithm is about 20-40% lower than the maximum link capacity, meaning that some energy saving is present. However, the reduction of the throughput of the TCP source, in the $TCP+PC$ case with respect to the $TCPonly$ case, is higher than in the previous cases, and because of this the loss probability becomes higher, so the action of the capacity scaling negatively affects the TCP performance. This performance degradation is higher for small



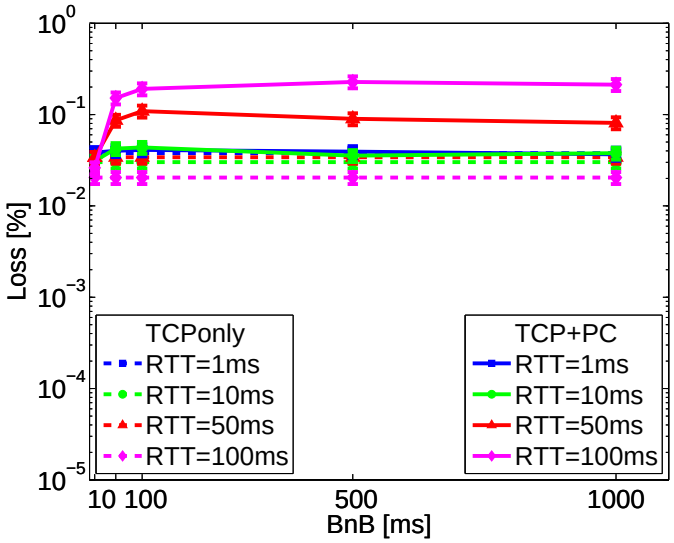
(a) Average Service Rate chosen by the capacity scaling algorithm



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



(c) Number of Service Rate changes



(d) Loss

Fig. 4. Background Traffic ($bON = 100ms$)

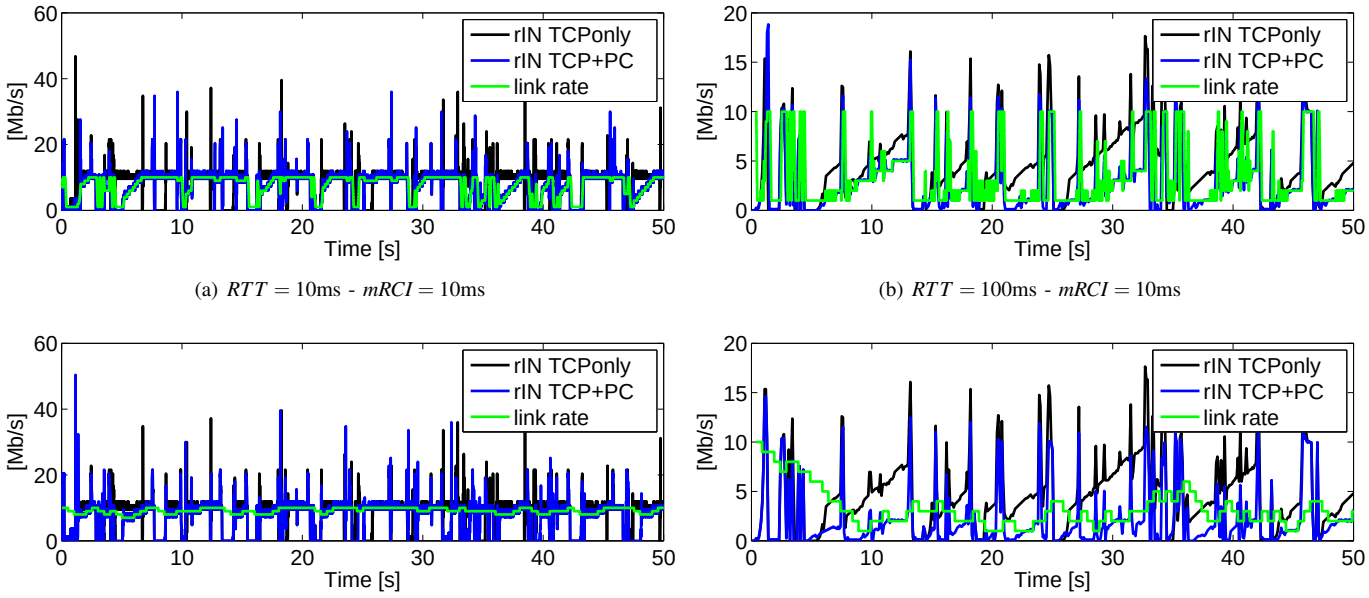
values of the *mRCI* parameter.

In order to understand the reasons of such results, Fig. 3 presents a comparison between the temporal evolution of the service rate chosen by the capacity scaling algorithm (green line) and the temporal evolution 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. Fig. 3(a) and Fig. 3(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 (Fig. 3(a)), the service rate exhibits small variations around the average value. Such variations disappear as the *mRCI* parameter becomes higher (Fig. 3(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 behavior to a non-capacity scaling mechanism.

On the contrary, when RTT is large (Fig. 3(b) and Fig. 3(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,

[2]The spikes in the figures are related to the increase of the TCP sending rate before a loss detection and the consequent sending rate reduction



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

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

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

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

Fig. 5. Background Traffic ($bON = 100ms$): 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

and the capacity scaling algorithm can reduce the service rate, thus providing some energy saving. However, as we noted before, in this 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 Fig. 3(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 seconds to reach again the 10Mb/s value in $TCPonly$ case, whereas it takes about 14 seconds in $TCP+PC$ case.

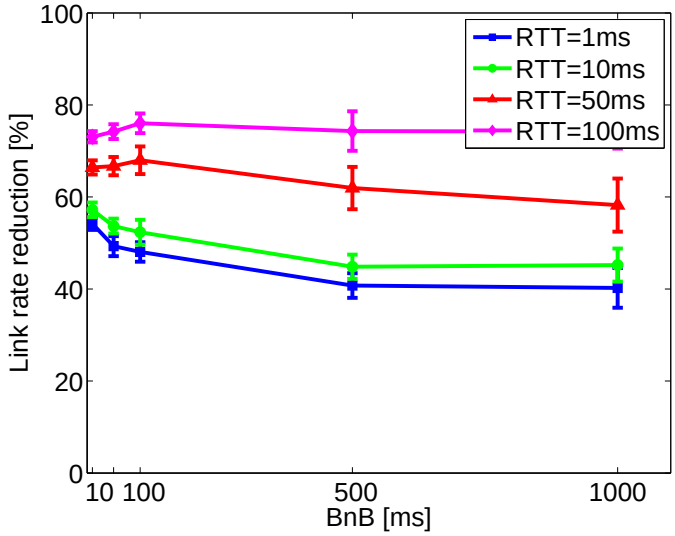
Let us now investigate the effects of larger values of the *mRCI* parameter, in the case of large RTT. As mentioned before, the performance degradation is lower when *mRCI* increases. However, note that the choice of a higher value of *mRCI* does not represent actually a better choice, as demonstrated in Fig. 3(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 of 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 (Fig. 3(b)). Therefore, we can conclude that the improvement of performance for increasing values of *mRCI*, is actually due only to the slower decrease of 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.

B. Background traffic

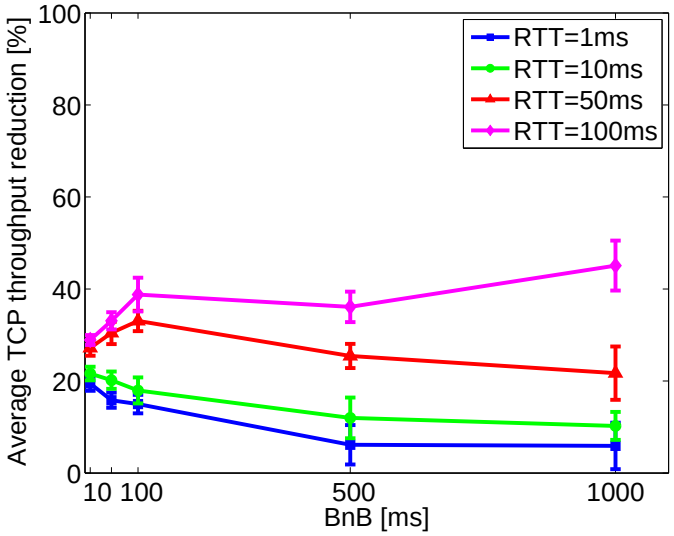
In this section we evaluate the impact of bursty background traffic on the interaction between TCP and capacity scaling mechanisms. We generated results for different dynamics of the background traffic. Confidence intervals at 95% confidence level were evaluated for all cases. For the sake of space,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

Fig. 4(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 behavior may assure some energy saving; however, as a consequence, the loss increases with respect to the $TCPonly$ case (Fig. 4(d)) and the reduction of the TCP throughput is huge, specially for large values of RTT (Fig. 4(b)).

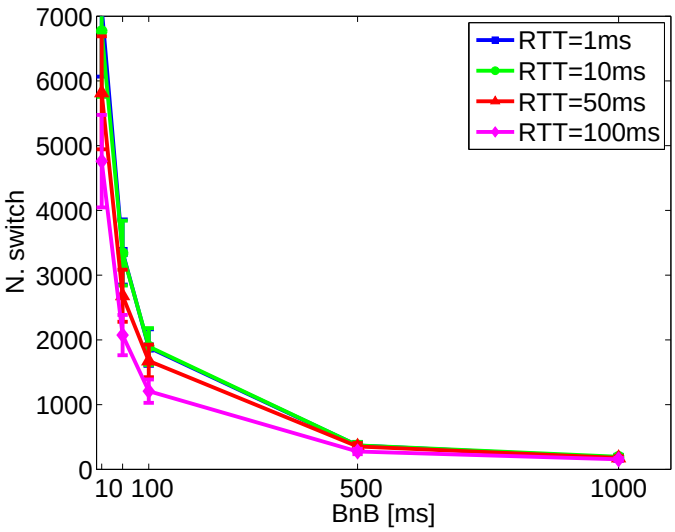
The reason for such results is that, in presence of bursty



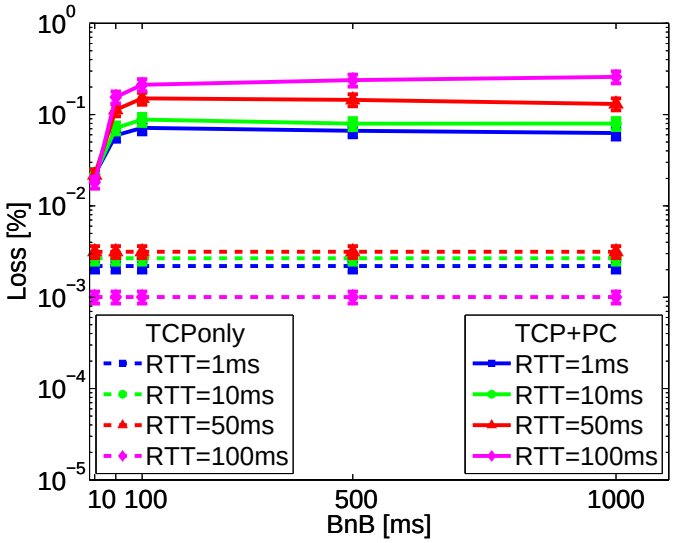
(a) Average Service Rate chosen by the capacity scaling algorithm



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



(c) Number of Service Rate changes



(d) Loss

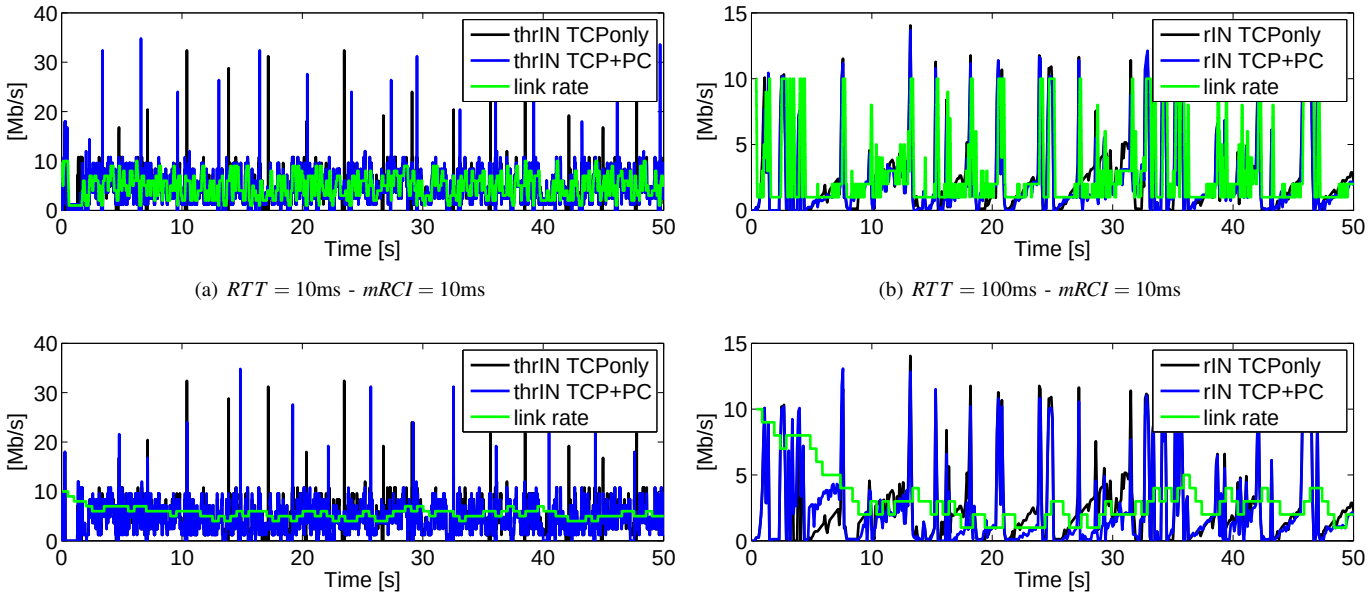
Fig. 6. Bottleneck ($BN = 50ms$) - Background Traffic ($bON = 100ms$)

background traffic, the variation of the arrival rate at node $N1$ becomes greater than for the case with only a single TCP source (compare Fig. 5 and Fig. 3). If the capacity scaling mechanism is slow (i.e. large values of $mRCI$), it may not notice in due time the increase of the arrival rate due to a spike of background traffic and the service rate may remain for long time lower than the arrival rate (Fig. 5(c) and Fig. 5(d)), forcing, in the meanwhile, losses and throughput reduction. The performance is even worse when large values of RTT are considered. 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 $mRCI$ 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 (Fig. 4(b) and Fig. 4(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 (Fig. 5(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.

C. Bottlenecks elsewhere

Finally, we analyzed the case in which a bottleneck is present elsewhere in the network, both in presence of a single TCP source, and in conjunction with background traffic. We have considered several dynamics for the variation of the



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

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

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

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

Fig. 7. Bottleneck ($BN = 50ms$) - Background Traffic ($bON = 100ms$): 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

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 Fig. 6 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 IV-B. 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 Fig. 4 with Fig. 6). 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 for example when comparing Fig. 7(b) with Fig. 5(b): when a bottleneck is present (Fig. 7(b)), 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 (Fig. 5(b)), 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 IV-B. 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 IV-B. Indeed, again, if the capacity scaling algorithm is too slow (i.e. for large values of $mRCI$), the need of 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. Fig. 7(d), 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 rate of the capacity scaling mechanism (see e.g. Fig. 7(b), around time instant 30). As a consequence, even achieving some energy saving (Fig. 6(a)) the performance degrades: the percentage of lost packets increases (Fig. 6(d)) and the TCP throughput is reduced (Fig. 6(b)).

V. Related work

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 for example [8], [9]). More recent works (see for example [10], [11]) look at the behavior of TCP in energy-efficient networks, 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 [4], [12]; 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 [5].

VI. 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 setups show that in a number of cases the two types of algorithms may interact in quite a negative fashion, with a drastic performance reduction of TCP flows. This is essentially due to the overlap of the two closed loop controls, with different time constants.

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.

References

[1] 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 Netweork Infrastuctures*, IEEE Communications Surveys & Tutorials, vol.13, no.2, pp.223-244, 2011.

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

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

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

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

[6] http://www.isi.edu/nsnam/ns/

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

[8] M.Zorzi, R.R.Rao, *Is TCP energy efficient?*, 1999 IEEE International Workshop on Mobile Multimedia Communications (MoMuC '99), pp.198-201, 1999.

[9] V.Tsaoussidis, H.Badr, X.Ge, K.Pentikousis, *Energy/throughput tradeoffs of TCP error control strategies*, Fifth IEEE Symposium on Computers and Communications (ISCC 2000), pp.106-112, 2000.

[10] A.Sassu, C.Scarso, F.Cuomo, *TCP behavior over a greened network*, Sustainable Internet and ICT for Sustainability (SustainIT 2012), pp.1-5, October 2012.

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

[12] C.Gunaratne, K.Christensen, B.Nordman, *Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed*, Int. J. Netw. Manag. 15, 5 (September 2005), 297-310.