Recovery Act:

Energy Efficiency of Data Networks through Rate Adaptation (EEDNRA)

Final technical Report

DOE Award Number: DE-EE0002887
Project Period: January 31, 2010 – March 31, 2011

Date of Report: June 30, 2011

Recipient Organization
Bell laboratories
Alcatel-Lucent Technologies
600 Mountain Avenue
Murray Hill, NJ 07974

Principal Investigator (PI)
Steven Fortune
Tel: (908) 582-7042
E-mail: Steven.Fortune@alcatel-Lucent.com

Team Members: N/A

Submitted to:

Industrial Technologies Program
Energy Efficiency and Renewable energy
U.S. Department of Energy

This report is based upon work supported by the US Department of Energy under Award DE-EE0002887.

1. EXECUTIVE SUMMARY 4

2. INTRODUCTION 5

3. BACKGROUND 6

4. RESULTS 6

4.1 MEASUREMENT OF CURRENT NETWORK-EQUIPMENT RATE-ADAPTIVITY 6
4.2 ALGORITHMIC METHODS TO OBTAIN RATE-ADAPTIVITY IN ROUTER PACKET PROCESSING 7
4.3 NETWORK-LEVEL ROUTING OF PACKET TRAFFIC TO MINIMIZE ENERGY 8
4.4 ESTIMATION OF POTENTIAL ENERGY BENEFITS OF RATE ADAPTIVITY IN SERVICE-PROVIDER NETWORKS. 8
4.5 REDUCING ROUTER BUFFER SIZES 9
4.6 PACKET-SCHEDULING ALGORITHMS WITH RATE ADAPTIVITY. 10

5. COMMERCIALIZATION 10

6. CONCLUSIONS AND BENEFITS ASSESSMENT 10

7. PUBLICATIONS AND PATENTS 11

8. REFERENCES 11

9. APPENDIX: ATTACHED PAPERS 12

1. **Executive Summary**

This Concept Definition Study focused on developing a scientific understanding of methods to reduce energy consumption in data networks using rate adaptation. Rate adaptation is a collection of techniques that reduce energy consumption when traffic is light, and only require full energy when traffic is at full provisioned capacity. Rate adaptation is a very promising technique for saving energy: modern data networks are typically operated at average rates well below capacity, but network equipment has not yet been designed to incorporate rate adaptation.

The Study concerns packet-switching equipment, routers and switches; such equipment forms the backbone of the modern Internet. The focus of the study is on algorithms and protocols that can be implemented in software or firmware to exploit hardware power-control mechanisms. Hardware power-control mechanisms are widely used in the computer industry, and are beginning to be available for networking equipment as well. Network equipment has different performance requirements than computer equipment because of the very fast rate of packet arrival; hence novel power-control algorithms are required for networking.

This Study was remarkably fruitful, resulting in five published papers, one internal report, and two patent applications, documented below. The specific technical accomplishments are the following:

- A model for the power consumption of switching equipment used in service-provider telecommunication networks as a function of operating state, and measured power-consumption values for typical current equipment.
- An algorithm for use in a router that adapts packet processing rate and hence power consumption to traffic load while maintaining performance guarantees on delay and throughput.
- An algorithm that performs network-wide traffic routing with the objective of minimizing energy consumption, assuming that routers have less-than-ideal rate adaptivity.
- An estimate of the potential energy savings in service-provider networks using feasibly-implementable rate adaptivity.
- A buffer-management algorithm that is designed to reduce the size of router buffers, and hence energy consumed.
- A packet-scheduling algorithm designed to minimize packet-processing energy requirements.

On the basis of this research, we believe that rate adaptivity offers an energy-savings opportunity, measured per unit of switching capacity, of at least a factor of two for service-provider networks, and possibly more for enterprise networks. This factor is orthogonal to and hence in addition to energy reductions resulting from improvements in semiconductor technology.

Additional research is recommended in at least two areas: further exploration of rate-adaptation in network switching equipment, including incorporation of rate-adaptation in actual hardware, allowing experimentation in operational networks; and development of control protocols that allow parts of networks to be shut down while minimizing disruption to traffic flow in the network.

The research is an integral part of a large effort within Bell Laboratories, Alcatel-Lucent, aimed at dramatic improvements in the energy efficiency of telecommunication networks. Some of the Bell Laboratories effort is part of Green

Touch [GT10], a telecommunications-industry consortium devoted to network energy efficiency.

This Study did not explicitly consider any commercialization opportunities.

2. Introduction

The Internet is growing at an exponential rate. According to a Cisco forecast [CISCO], traffic is growing at a rate of roughly 30% per year, depending upon region. By 2015, total network traffic is estimated to be nearly a zettabyte (10^21 bytes) per year, and there is every reason to believe that growth will continue after that. Though the use of data networking often replaces other carbon- and energy-intensive activities (e.g. videoconferencing substitutes for travel [S20]), this exponential growth raises concern about the energy consumption of networking.

From the perspective of a telecommunications service provider such as Verizon or AT&T, network infrastructure can be split into wireless access (e.g. the cellular radio infrastructure that communicates with mobile devices), fixed or wireline access (which provides wired connectivity to homes and businesses), core switching, and long-distance optical data transport. The research summarized here concerns the energy efficiency of core switching, the IP routers and switches that provide packet-based data services. Such routers and switches are also used in enterprise networks, contained within a single corporate, academic, or government institution.

The approach is to make the packet network rate-adaptive, that is, to make electricity consumption depend upon traffic, increasing as traffic levels increase. There are two motivations for this: first, as documented below, current packet switching equipment is at best minimally rate adaptive—energy consumption varies only slightly with traffic. Second, networks typically have a provisioned capacity that is much larger than average traffic. For service-provider networks, overprovisioning is required because of the large daily fluctuations in traffic and the redundancy required to handle equipment failures; for enterprise networks, where traffic is not aggregated to the same extent as service-provider networks, overprovisioning is required to handle short-duration traffic bursts.

Rate adaptivity results from the algorithms and protocols that are implemented in software or firmware, exploiting current and expected hardware capabilities that control energy usage. Such algorithms should be applicable across a broad spectrum of network hardware, irrespective of specific manufacturer or implementation technology. Energy savings obtained by algorithms are orthogonal to and hence in addition to energy savings obtained by improvements in component technologies, e.g. improvements in complementary metal-oxide semiconductor (CMOS) technology for integrated circuits.

The hardware capabilities required to control energy usage are well-established in the computer industry. For example, sleep-state exploitation allows a processor to be put into a sleep state that uses very little power but allows no processing. Similarly, dynamic voltage-frequency scaling allows a broad range of control of processing rate, though with power consumption potentially growing cubically with rate. However these techniques have not yet been well-exploited in the networking industry. One difficulty, for example, is that the time to change processor state, e.g. in or out of sleep mode, is large compared with the time interval between packet arrivals in a router.

The research that was performed using the funding provided by this Department of Energy grant falls into several conceptual categories, each described in more detail below.

1. We measured experimentally the rate-adaptivity of typical current network switching equipment, to provide a baseline for comparison with future equipment.
2. We considered rate-adaptation at the level of an individual network element. In particular, we explored control algorithms for use in routers to obtain rate adaptivity, while still respecting performance requirements like delay minimization.
3. We considered rate adaptation at the global network level. In particular, we developed algorithms that reroute traffic to minimize energy consumption, assuming a model of router energy use.
4. We estimated the total energy savings that can be obtained by thee two optimizations described in (2) and (3).
5. We developed buffer-control algorithms designed to dramatically decrease the amount of memory required by a router, enhancing its rate-adaptivity since the energy required for buffer memories is largely independent of traffic.
6. We explored the interaction of rate adaptation with scheduling, and considered approaches that further minimize energy consumption by appropriate scheduling of packet processing.

3. Background

Rate adaptation has been widely considered in the computing community [BH07, BPSB00, CSBETW08] but is only recently being addressed in the networking community. Chabarek et al [CSBETW08] report results related to 4.1 and 4.4 below; they measured the energy consumption of switching equipment in a variety of scenarios and used the constructed energy models with combinatorial optimization techniques to minimize energy use in simulated networks. Nedevschi et al [NPRIW08] consider rate adaptation for network elements, related to 4.2 below; they discuss algorithms that batch packets into larger bursts, to mitigate the overhead required for processor state changes, and analyze the size of queues using rate adaptation. The Internet Engineering Task Force (IETF) is developing standards for Ethernet links that allows for very rapid state changes from sleep mode to active [IETF803].

We have also published preliminary results related to the material in this report. Francini et al [FS10, FS10a] consider router packet-processor control algorithms using sleep-state-exploitation and dynamic frequency-voltage scaling (DVFS), based on a simplified router model. Andrews et al [AFZZ10] analyze the intrinsic computational difficulty of routing to minimize energy, with an energy model derived from CMOS power-frequency curves.

4. Results

The research supported by this Department of Energy grant is briefly summarized below. Each of the sections 4.1 through 4.6 corresponds to one of the papers [1] through [6] cited below.

4.1 Measurement of current network-equipment rate-adaptivity

To establish characteristics of current technology, and to provide a baseline for future improvements, the electrical power consumption of Internet Protocol (IP) switching equipment was measured in a variety of scenarios. Three different equipment types were measured: an Ethernet switch in a fixed configuration (no plug-in cards), an aggregation router in a fixed configuration, and an aggregation router in a modular configuration, with 8 slots for plug-in cards. These equipment types are representative of switching equipment currently deployed in the access part of a service-provider network.

Equipment details, measurement methodology, test configuration, and detailed results appear in [1]. A summary statement is that this equipment is not significantly rate-adaptive: the variation in electricity consumption from no load to full load is at most about 6%. The actual variation depends upon the exact configuration, and in fact was usually only a couple of percent.

A surprise from the measurement was the magnitude of the change in electricity consumption due to `enabling' an interface, that is, using the software management capability to switch an interface from `powered-on but refusing traffic' to `powered-on and accepting traffic'. Enabling an interface increases electricity consumption more than a traffic variation from no load to full load. This effect was most significant for the Ethernet switch, where the combination of enabling the interface plus processing traffic represented 10-20% of total electrical consumption; for the two routers the combination was less than 10%.

4.2 Algorithmic methods to obtain rate-adaptivity in router packet processing

At a high level, a router must perform two significant functions, packet processing and switching. Packet processing consists of examining the header of an individual packet, to determine the destination of the packet and the priority with which it should be handled. Once the destination is known, switching routes the packet through the internal switch fabric of the router to the appropriate external interface. We estimate that packet processing requires 20-50% of the electrical consumption of routers, and the switch fabric somewhat less.

Modern packet processing is quite complex because of the many available protocols that allow administrators to set the policy for management of networks. Packet processing is performed by a network processor, much like the CPU in a laptop or desktop workstation. Two techniques have been developed in the computer industry for energy management of laptops and desktops: sleep-state exploitation, which puts the CPU into a low-energy sleep mode when there is no processing to be performed, and dynamic frequency-voltage scaling (DVFS), which allows the processing rate to be increased at the expense of increased energy consumption.

It would be desirable to use these two techniques in a network processor, so that the energy consumption of packet processing would be rate-adaptive, with low consumption in periods of light load. A difficulty is the difference in time scales between packet processing and processor state-changing: at a typical 10 Gbps (10 gigabits per second) a packet may arrive every microsecond, even in times of light load, whereas changing state, e.g. to and from sleep mode, may take hundreds of microseconds.

The algorithmic issue is to choose a control mechanism that minimizes energy consumption while also minimizing the delay in processing packets. Minimizing per-router packet delay is important because a packet may traverse tens of routers in an end-to-end path in a network, and excessive delay degrades the quality of real-time applications such as video or voice.

Prior work [FS10] considers a simple model of a router as a single `rate adaptation domain' consisting of a buffering stage where packets can be stored temporarily and a processing stage that is subject to state-changing for energy management. The conclusion of the prior work was that it was beneficial to use both techniques, sleep-state exploitation and DVFS, in combination. The new work here [2] considers a more realistic model of a router consisting of multiple consecutive rate-adaptation domains. In this scenario, the conclusion is that sleep-state exploitation by itself gives most of the energy saving benefit, and DVFS provides little in addition. Use of only the single technique simplifies the overall control mechanism.

4.3 Network-level routing of packet traffic to minimize energy

As just described, there are techniques that can be used to make a router rate-adaptive. However, it is unlikely to be feasible to make a router perfectly rate-adaptive, that is, to have a completely linear relationship between traffic and electricity consumption, with zero consumption at zero traffic. This is because of the overall complexity of the router, with various components (e.g. lasers used at the physical layer) that cannot be made rate adaptive. The expectation is that a router will always have some start-up power consumption when it is turned on, even without any traffic. It may be possible that each interface on a router can be individually powered on or off, and hence each interface has an individual start-up power in addition to a common start-up power for the router as a whole.

An alternate approach to rate-adaptation is network-wide routing to minimize energy consumption. At times of light load, traffic can be concentrated on some links in the network, allowing some routers or interfaces on routers to be powered down completely, saving the start-up power.

The work in [3] addresses the algorithmic problem of how to route traffic in a network to minimize energy consumption. The problem is stated in terms of a network given as a graph, a traffic matrix that specifies source, destination, and required capacity of each traffic demand, and a description of the traffic-energy curve of each router that specifies energy as a function of traffic. The traffic-energy curve is assumed to be zero at zero traffic, has a fixed start-up cost, and then increases with traffic. This algorithmic problem is known to be computationally difficult (NP-hard) [AFZZ10], so the new result is a computationally efficient (polynomial time) algorithm that approximates the minimal-energy routing rather than finding it exactly. This research demonstrates the scientific possibility of an efficient approximation algorithm, but more refinement would be necessary for the algorithm to be useful in a practical setting.

4.4 Potential energy benefits of rate adaptivity in service-provider networks.

The two techniques just described, per-router rate adaptation and network-level routing, address the same opportunity for energy efficiency. Both try to save energy in cases where traffic is low. However both require substantial engineering effort to implement, in one case requiring redesign and reprogramming of routers, and in the other case changing network routing protocols and management tools.

The study reported in [4] attempts to estimate, as realistically as possible, the total opportunity for energy savings in service-provider networks using the two techniques, and in particular to estimate whether one technique alone would be sufficient or there would be additional benefit using both techniques. The approach used is to simulate the way a service provider would design a network, for example to handle peak traffic and to provide redundancy in the network to handle possible equipment failures. Then the relative amount of energy used by the network was determined in a variety of scenarios. Total traffic was varied from 10% to 100% of capacity, both with and without energy-aware routing; various router traffic-energy curves were used, varying the startup cost from 0%, i.e. perfect rate-adaptivity, to 100%, completely non-rate-adaptive.

The conclusion of the study was that under plausible assumptions about what can be implemented--with per-router rate adaptation (50% startup cost), network-level routing, and the observed daily fluctuations in traffic (three-to-one peak-to-valley fluctuation)--the total energy required for routers in a service-provider network can be reduced by roughly a factor of 2. This is compared with the current technology of non-rate-adaptive routers and energy-oblivious routing. Both per-router rate adaptation and network-level routing contribute to the improvement, depending upon the exact level of assumed startup cost.

4.5 Reducing router buffer sizes

Routers typically have large buffer memories that are used to store packets during periods of traffic congestion, before they can be processed and routed to the appropriate outgoing interface. The memory, typically implemented with dynamic random access memory (DRAM), constitutes a significant part of the energy consumption of a router, estimated at 10-20%. It is hard to make buffer energy consumption rate-adaptive, so it would be of benefit to minimize the size of the required buffer memory.

Traffic congestion in the Internet is managed by an interaction between the transmission control protocol (TCP) and the packet drop algorithms used by routers. TCP is used at the endpoint of a traffic flow (e.g. in a laptop or web server) to determine the rate at which the endpoint sends traffic. In normal operation TCP continuously attempts to increase traffic rate until traffic congestion results. Congestion is detected if a packet is dropped in the network, at which point the host reduces traffic rate. A packet can be dropped by a router because of buffer overflow or as an intentional explicit signal of congestion.

The interaction between TCP and packet drop by routers is complex and has been extensively studied. Routers are typically configured with buffers that can handle a few hundred milliseconds of traffic; the goal is to keep links utilized even if there are oscillations in traffic flow resulting from the control loop between TCP and router packet drop.

Reducing buffer size requires better algorithms that determine when to drop packets as a contention signal. A well-known proposal [FJ93] is Random Early Detection (RED), which drops packets randomly at a rate that increases with buffer utilization, and hence incipient congestion. The goal is to signal congestion across flows but avoid global traffic oscillations. The new work in this study [5] proposes a new algorithm, called Periodic Early Drop (PED), which is an extension of RED. PED has a slower control loop than RED, which further minimizes traffic oscillations, which remain with RED. In simulations, PED allows buffers to be reduced in size by

two orders of magnitude from current levels while still maintaining full link utilization.
This potentially allows a significant reduction in the non-rate-adaptive energy consumption of a router.

4.6 Packet-scheduling algorithms with rate adaptivity.

As mentioned above, the adjustment of processing rate using dynamic voltage-frequency scaling (DVFS) is an important tool for rate adaptation. One consequence of DVFS is that at high processing rates, the energy used per operation is more than at low processing rates. Hence it is better to run at slow rates, as long as performance criteria can be met.

The new research in this study [6] uses this observation to adjust the packet processing rate in a router. If the packet-processing work queue in a router is long, then an obvious strategy is to increase the processing rate to empty the queue. However, this may not be the energy-optimal choice, for example, if no more packets will arrive in the near future, then it may suffice to process the packets at a lower rate. The algorithm in [6] adjusts the packet processing rate to be nearly energy and queue-length optimal, even without advance knowledge of the packet arrival pattern.

5. Commercialization

This project was a Concept Definition Study and did not explicitly consider commercialization. Alcatel-Lucent is a leading vendor of IP switching equipment, and hence is in a position to exploit the research described here.

6. Conclusions and Benefits Assessment

Rate adaptation is a promising technique for improving the energy efficiency of networks of routers and switches. We believe that the plausible energy-saving opportunity is least a factor of 2 for service-provider networks, as they are currently designed and operated, and perhaps even a larger factor for enterprise networks, which are even more over-provisioned. This factor is orthogonal to and in addition to improved efficiency that results from improvements in semiconductor technology. It is difficult to estimate the absolute value of possible savings, because the two trends of increasing traffic and semiconductor improvements are both very rapid but affect the possible savings in opposite ways.

7. Recommendations for Future Work

There are at least two important areas for future research. One is to actually build switching equipment, such as a router, which incorporates rate adaptivity as an integral part of its design. The packet-processor control algorithms explored in this study are worthy of consideration. In addition, there are many VLSI-level hardware optimizations that can be considered as well, e.g. stopping the internal hardware clock during a temporary idle period, or powering down parts of a chip during a longer-term idle period. All of these considerations compete with the usual design challenges, e.g. minimizing cost, supporting the complex set of features required in a modern router, and meeting performance and reliability goals. Once switching equipment with rate-adaptivity is built, such equipment can be used in real network environments to evaluate actual energy savings.

The second important research are is to investigate the hardware support and protocol changes required to allow networks to be dynamically configured to adapt to changing traffic. For example, one requirement is that a router or router interface can be shut down and then restarted quickly while regaining its control state—it is not clear that this capability exists or how easily it can be implemented. A second requirement is that network routing algorithms adapt quickly and robustly to changes in the network, as routers and interfaces shut down and restart. Network operators are unlikely to tolerate disruptions to traffic flow, e.g. packet loss or delay. Again it is unclear whether current protocols are sufficiently robust. These issues require further study.

7. Publications and patents

The research reported in the following papers was supported in part by the EEDNRA grant.

[1] M. Ricca, A. Francini, S. Fortune. Energy Profiling of Network Equipment. Unpublished manuscript, Alcatel-Lucent, 2011.

[2] A. Francini. Selection of a Rate Adaptation Scheme for Network Hardware. Submitted, Alcatel-Lucent, 2011.

[3] M. Andrews, S. Antonakopoulos and L. Zhang. Minimum-Cost Network Design with (Dis)economies of Scale. *Proceedings of the 51st Annual IEEE Symposium on Foundation of Computer Science (FOCS)*. Las Vegas, NV, October 2010.

[4] S. Antonakopoulos, S. Fortune and L. Zhang. Power-aware Routing with Rate Adaptive Networking Elements. *Proceedings of the 3rd International Workshop on Green Communications.* Miami, FL, December 2010.

[5] Andrea Francini. "Beyond RED: Periodic Early Detection for On-Chip Buffer Memories in Network Elements," to appear in 2011 *IEEE Conference on High-Performance Switching and Routing (HPSR 2011)*, Cartagena, Spain, July 2011.

[6] M. Andrews, S. Antonakopoulos and L. Zhang. Energy-aware Scheduling Algorithms for Network Stability. *Proceedings of IEEE INFOCOM 2011*. Shanghai, China, April 2011.

The following have been filed with the US Patent office:

M. Andrews, S. Antonakopoulos and L. Zhang. Energy-Efficient Network Devices with Coordinated Scheduling and Processing Rate Control. Serial No. 13/078,599.

A. Francini, "System and Method for Implementing Periodic Early Discard in On-Chip Buffer Memories of Network Elements." 807985-US-NP.

8. References

[AFZZ10] M. Andrews, A. Fernadnez, L.Zhang, W.Zhao. Routing for Energy Minimization in the Speed Scaling Model. *IEEE INFOCOM 2010*.

[FS10] A. Francini, D. Stiliadis. Performance bounds of rate-adaptation schemes for energy-efficient routers, *Proc. of IEEE HPSR 2010*.

[FS10a] A. Francini, D. Stiliadis. Rate adaptation for energy efficiency in packet networks. *Bell Labs Technical Journal,* 2010, pp. 131--146.

[FJ93] S. Floyd, V. Jacobson. Random Early Detection (RED) gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking*, August 1993.

[CISCO] Cisco Visual Networking Index: Forecast and Methodology, 2010-2015, available on www.cisco.com.

[BH07] L.A. Barroso, U. Holze. The Case for Energy-Proportional Computing, *IEEE Computer,* Volume 40, Issue 12, pp. 33–37, December 2007.

[BPSB00] T. Burd, T. Pering, A. Stratakos, R. Brodersen, A Dynamic Voltage Scaled Microprocessor System, *IEEE Journal of Solid-State Circuits*, 35:11, November 2000.

[CSBETW08] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, S. Wright. Power Awareness in Network Design and Routing, *Proceedings of IEEE Infocom 2008*, pp. 1130-1138, April 2008.

[IEEE802] IEEE P802.3az Energy Efficient Ethernet Task Force http://www.ieee802.org/3/az/

[INTEL04] Enhanced Intel® SpeedStep® Technology for the Intel® Pentium® M Processor, White Paper, March 2004, ftp://download.intel.com/design/network/papers/30117401.pdf.

[GT10] Green Touch consortium, http://www.greentouch.org/

[NPIRW08] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, D. Wetherall. Reducing Network Energy Consumption via Sleeping and Rate-Adaptation, *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2008)*, San Francisco, CA, pp. 323–336, April 2008.

[S20] The Climate Group, "SMART2020: Enabling the low carbon economy in the information age (http://www.smart2020.org).

9. Appendix: attached papers

US DOE Project:

Energy Efficiency of Data Networks Through Rate Adaptation (EEDNRA)

Task 1:

Energy Profiling of Network Elements

Authors: Marco Ricca, Andrea Francini, Steven Fortune

Document Version: 0.203

Updated: 07.09.2011

Abstract

In packet networks, rate adaptation technologies aim at establishing a direct, possibly linear relationship between power consumption and traffic load. Since such technologies are typically not deployed in commercial networks as of 2010, it is useful to assess how amenable existing network elements are to their immediate introduction and identify the design upgrades that can maximize their energy savings in new generations of network systems. The formulation of energy profiles for sample commercial equipment is a first step in this direction.

In this document we describe how we derive the energy profiles of three commercial network systems, namely one compact Ethernet switch for edge and enterprise applications and two routers for service aggregation in access networks. The profiles are based on a novel formulation of the linear model that defines the overall power consumption of the system as a function of the operating states of all of its components (chassis, line cards, and ports). The results of our experiments indicate that current equipment is not well suited to flow-timescale rate adaptation techniques that remotely control the operating state of parts of a system to minimize the overall power consumption of the network. To create a better match, software and hardware upgrades should be made in system designs to support low-power sleep states for unutilized system components, and to enable packet-timescale rate adaptation methods that instantly adjust the power consumption of data-path hardware to the rate profile of the input traffic.

Table of Contents

1. Introduction 1
2. Systems under Test 2
  2.1 Overview of SUT1 Service Capabilities 3
  2.2 Overview of SUT2 and SUT3 Service Capabilities 3
3. Energy Profiling 3
  3.1 Review of Previous Work 3
    3.1.1 Chabarek et al., 2009 3
    3.1.2 Mahadevan et al., 2009 4
    3.1.3 Tamm et al., 2010 5
    3.1.4 ECR Initiative, 2010 5
  3.2 Reference Model for Energy Profiling 5
4. Experimental Testbed 6
  4.1 Definitions and Units 7
  4.2 Laboratory Equipment 7
  4.3 Testbed Setups 10
    4.3.1 SUT1 Reference Testbed Setup 10
    4.3.2 SUT2 and SUT3 Reference Testbed Setup 10
5. Power Measurements Methodology 11
  5.1 Power Consumption Estimation for SUT2 and SUT3 11
  5.2 Estimation of Port Parameters 14
  5.3 Service Configuration Constraints in SUT2 and SUT3 15
  5.4 Identity of Input and Output Saturation Power Values 15
6. Experimental Results 16
  6.1 Energy Profile of SUT1 16
    6.1.1 Summary of SUT1 Results 16
    6.1.2 Idle Chassis Power 17
    6.1.3 Idle Line Card Power 17
    6.1.4 Port Power, 1GbE Port with 1000BASE-TX SFP 17
    6.1.5 Port Power, 1GbE Port with 1000BASE-SX SFP 17
    6.1.6 Port Power, 1GbE Port with 1000BASE-LX SFP 17
    6.1.7 Port Power, 10GbE Port with 10GBASE-LW/LR XFP 17
    6.1.8 Description of SUT1 Experiments 18
  6.2 Energy Profile of SUT2 23

6.2.1 Summary of SUT2 Results 23
6.2.2 Idle Chassis Power 23
6.2.3 Idle Line Card Power 23
6.2.4 Port Power, 10/100BASE-TX Ethernet Port 23
6.2.5 Port Power, 1GbE Port with 1000BASE-TX SFP 23
6.2.6 Port Power, 1GbE Port with 1000BASE-SX SFP 24
6.2.7 Port Power, 1GbE Port with 1000BASE-LX SFP 24
6.2.8 Description of SUT2 Experiments 24
6.3 Energy Profile of SUT3 28
6.3.1 Summary of SUT3 Results 28
6.3.2 Idle Chassis Power 29
6.3.3 Idle Ethernet Line Card Power 29
6.3.4 Port Power, 10/100BASE-TX Ethernet Port 29
6.3.5 Port Power, 1GbE Port with 1000BASE-TX SFP 29
6.3.6 Port Power, 1GbE Port with 1000BASE-SX SFP 29
6.3.7 Port Power, 1GbE Port with 1000BASE-LX SFP 29
6.3.8 Description of SUT3 Experiments 30
7. Possible Refinements of the Model for Energy Profiling 34
7.1 Effects of Packet Size 35
7.2 Non-Linear Behaviors 36
7.2.1 SUT1 Cooling Fan 36
7.2.2 SUT2 and SUT3 Throughput Cap 36
8. Conclusions 37
8.1 Energy Profiling Methodology 37
8.2 Rate Adaptation in Network Elements 37
9. References 38

1. *Introduction*

In packet networks, the term rate adaptation designates a broad set of technologies that aim at establishing a direct relationship between sustained workload and energy consumption. Ideally, the energy-workload function should be linear, with minimum gradient and no energy consumed in the absence of packet traffic [Barroso]. To approximate such behavior, rate adaptation schemes typically provide the systems that they control with a discrete set of operating states, where each state maps a fixed traffic processing rate onto a respective power consumption level.

The scope of the control exercised by a rate adaptation scheme can range from large subsets of network links and nodes [Anonak], [Rossi] to individual sections of a single traffic processing chip [Benini]. Hence, for the sake of clarity we find it necessary to partition rate adaptation techniques based on their timescale of operation, which is defined by the switching time needed to transition between states and ultimately depends on the size of the targeted system. Flow-timescale rate adaptation (FTRA) techniques control the state of network links and nodes based on expected or measured trends in traffic demands between network endpoints [Antonak], [Rossi]. FTRA state transitions involve network signaling and system-level power cycles, so their timescale ranges from seconds to minutes. Packet-timescale rate adaptation (PTRA) techniques adjust the clock frequency and supply voltage of data-path hardware components to locally maintained workload indicators such as queue lengths and traffic arrival rates [Nedevschi], [Francini]. The timescale of PTRA state transitions ranges from microseconds to milliseconds depending on the underlying integrated circuit technology. Bit-timescale rate adaptation (BTRA) also applies to data-path hardware components. Compared to PTRA, BTRA transitions are much faster to execute (down to nanoseconds) because they only involve control of the system clock (e.g., by gating of the clock signal), at the expense of reduced power savings.

Task 1 of US DOE project EEDNRA, named “Energy Profiling of Network Equipment,” relies on power consumption measurements to obtain the energy profiles of network systems that are commercially available as of 2010. The energy profiles make it possible to assess the energy-saving benefits that may derive from the application of rate-adaptation techniques to current-generation network systems. In particular, in the case of network-wide FTRA techniques the profiles quantify the benefits of enabling and disabling line cards and network ports based on the traffic load to be sustained; in the case of PTRA and BTRA techniques, instead, the profiles identify the energy-saving margins that are available for the introduction of rate-adaptive hardware components.

The energy profile of a network element maps system and traffic configurations onto power consumption levels, typically by means of a simplified linear model. Examples of system configuration variables that make up an energy profile include the number of cards plugged into the chassis (in modular systems), the number of network ports that are attached to a network link, the number of connected ports that are enabled for network operation, and the transmission capacity provisioned for the ports that are enabled for network operation. Traffic configuration variables include the traffic arrival rate at each network port and the statistical distribution of packet sizes and packet inter-arrival times at ports where traffic is present.

While energy profiles are commonly available for computing systems and processors, studies that focus on networking systems and components have started appearing in the literature only recently [Chabarek], [Mahadevan], [Tamm], and always with important limitations. In fact, the energy profiles presented in those studies are either too coarse [Chabarek], or based exclusively on manufacturer power-rating data rather than experimental measurements [Tamm], or inaccurate in the modeling of system components [Mahadevan]. In Task 1 of project EEDNRA we construct linear energy profiles for three commercial network systems, namely one layer-2 switch and two layer-3 routers. As opposed to the prior studies that we are aware of [Chabarek], [Mahadevan], [Tamm], we derive fine-grained energy profiles from experimental measurements based on system configurations that are common in commercial applications.

The contribution of this work is twofold. First, we define a methodology for the experimental construction of energy profiles that introduces clear elements of novelty as compared to the prior art. Second, we collect results that suggest that the systems under study lack the hardware ability to control power levels and hence that network-wide FTRA techniques would only yield marginal energy savings. Only the introduction of PTRA/BTRA capabilities in their data-path hardware components can make those savings more substantial.

The document is organized as follows. In Section 2 we describe features of the systems under test that are relevant to the formulation of the energy profiles and configuration of the measurement experiments. In Section 3 we define our reference model for energy profiling after reviewing prior work in the same space. In Section 4 we describe components and configurations of our experimental testbed. In Section 5 we present the methodology that we follow for execution of the power measurements, including choices that we have to make to address issues deriving from the lack of instruments for direct measurement of certain metrics. In Section 6 we illustrate the results of our power measurement experiments. In Section 7 we illustrate further refinements of our reference model for energy profiling which, although promisingly viable, will need future validation from more comprehensive measurements. Finally, in Section 8 we summarize the results of our work.

2. Systems under Test

We obtain energy profiles for the following three network systems:

- Ethernet switch in fixed system configuration with integrated control and switch module (no slots for plug-in cards), twenty-four (24) 10/100/1000 Ethernet (SFP) ports, two (2) 10Gbps Ethernet (XFP) ports. AC power supply: $100-240\,V,\ \ 50-60\,Hz$.
- Aggregation router in fixed system configuration with integrated control and switch module, six (6) 10/100 Ethernet (RJ-45) ports, two (2) 10/100/1000 Ethernet (SFP) ports, sixteen (16) T1/E1 ports (not used in our experiments). DC power supply: two feeds at $-48V$.
- Aggregation router in modular system configuration with 8-slot chassis. In the SUT3 instance available for our experiments, the chassis is populated with one (1) fan card, two (2) Control and Switch Module (CSM) cards, two (2) 8-port Ethernet adapter cards (EAC's). Each Ethernet adapter card includes six (6) 10/100 Ethernet (RJ-45) ports and two (2) 10/100/1000 Ethernet (SFP) ports. DC power supply: two feeds at $-48V$.

Note: Ethernet (RJ-45) identifies an integrated 10/100BASE-TX Ethernet port. Ethernet (SFP) identifies an Ethernet port that accommodates a small form-factor pluggable (SFP) optics module. The SFP itself can be of different types depending on the type of cable connector that it supports: 1000BASE-LX and 1000BASE-SX SFP's support optics cables, 1000BASE-TX SFP's support copper cables with RJ-45 connectors. Ethernet (XFP) identifies a $10\,Gbps$ Ethernet (10GbE) port that accommodates a 10GBASE-LW/LR small form-factor pluggable optics module (XFP).

The service capabilities of the three SUT's are illustrated with more detail in the following subsections.

2.1 Overview of SUT1 Service Capabilities

SUT1 is a compact Ethernet-edge and aggregation switch with MPLS capabilities. It can be used for establishment of Carrier Ethernet VPN services such as virtual private LAN service (VPLS), hierarchical VPLS (H-VPLS), virtual leased line (VLL) service, and Ethernet access to enhanced Internet services (IES) and IP VPN services.

2.2 Overview of SUT2 and SUT3 Service Capabilities

SUT2 is the compact, low-power member of a family of multiservice adaptation and aggregation routers.

SUT3 is a larger, modular version (8 slots) of the same router. It offers an extended set of service combinations when adapter cards of different types are installed. Two of the eight slots are populated with CSM cards, the bottom-left slot with an Ethernet adapter card.

The product family of SUT2 and SUT3 is suited to the aggregation and backhaul of 2G, 3G and LTE mobile traffic in the IP/MPLS radio access network (RAN) and to applications for enterprises, energy utilities, and transport and government agencies. The family supports Ethernet pseudo-wire encapsulation per port and per VLAN, IP pseudo-wires, BGP/MPLS virtual private networks for segregation of layer-3 traffic, IP routing and forwarding, and native switching of ATM, Ethernet and TDM traffic, all with advanced traffic management capabilities.

3. Energy Profiling

This section discusses possible ways of defining energy profiles for network equipment. We start with proposals that have appeared in the literature in the last few years and conclude with the description of the approach that we use for execution of Task 1 of project EEDNRA.

3.1 Review of Previous Work

3.1.1 Chabarek et al., 2009

In [Chabarek], Chabarek et al. construct energy profiles for two IP routers manufactured by Cisco Systems, namely the GSR 12008 core router and the 7507 edge router. The GSR 12008 chassis contains twelve slots, of which one accommodates the route processor card and two others are dedicated to switch fabric modules (10Gbps capacity each). The remaining nine slots can be used for network adapter cards, each with capacity up to 4Gbps. The 7507 has a seven-slot chassis, with one slot for the route processor card and the other six slots for adapter cards, each with capacity up to 1Gbps.

A first set of experiments yields for each system the power contribution of the chassis and of the different types of modules that can be installed in the adapter slots. No cables are attached to the network interfaces, so obviously no traffic flows through the system. The measured power consumption is the bare sum of the chassis and card contributions in idle state.

A second set of experiments focuses on the GSR 12008, in a configuration that includes the route processor card, the switch fabric cards, one adapter card with four 1Gbps Ethernet interfaces (of which only three connected), and one adapter card with one OC-48 interface, also connected. Traffic flows from the three 1Gbps Ethernet interfaces to the OC-48 link. The experiments focus on the power consumption effects of different types of traffic (CBR versus bursty TCP traffic), of different packet sizes (100, 576, and 1500 bytes), of different routing table sizes, and of different routing functions. Overall, the power contribution that derives from the presence of traffic is relatively small compared to the total, showing that the introduction of BTRA/PTRA capabilities could provide important energy-saving benefits. The differences observed across different traffic configurations are even smaller.

The following equation defines the linear model that the authors of [Chabarek] adopt for construction of their energy profiles:

$$P_S = P_C + \sum_{i=1}^{N_L} \left(P_{L,i} + T_{L,i}(\gamma_i) \right), \tag{1}$$

where P_C is the power consumed by the chassis when idle, N_L is the number of line cards, possibly of different types, that are plugged into the chassis, $P_{L,i}$ is the power consumed by a line card i when idle, and $T_{L,i}(\gamma_i)$ is the additional power contribution of the same line card when traversed by traffic at load γ_i $(0 \le \gamma_i \le 1,$ where $\gamma_i = 1$ when line card i is fully loaded).

Since it is primarily utilized in the optimization of network planning decisions, the model of (1) does not try to single out the power contribution of individual network ports, especially when connected to a network cable but not enabled for handling traffic. This power contribution should always be included in the energy profiles that drive the operation of network-wide FTRA techniques, so that the effects of selectively controlling the state of individual links out of a line card are properly accounted for.

3.1.2 Mahadevan et al., 2009

The study presented by Mahadevan et al. in [Mahadevan] has strong elements of similarity with our work, with respect to both end goals and measurement methodology.

The authors obtain energy profiles for seven distinct systems of different sizes and capabilities, including an Ethernet hub, a wireless access point, three edge LAN switches, one core switch, and one edge router. The following equation defines the reference model for construction of the energy profiles:

$$P_S = P_C + \sum_{i=1}^{N_L} P_{L,i} + \sum_{j=1}^{N_P} P_{P,j} \cdot \rho_j \,, \tag{2}$$

where N_P is the total number of ports that are connected and enabled, $P_{P,j}$ is the power consumed by a port j when fully loaded, which depends on the type of the port and on its configuration, and ρ_j is the traffic load sustained by the port $(0 \le \rho_j \le 1,$ with $\rho_j = 1$ when the port is fully loaded). We remark that (2), although expressed in a different form, is fully equivalent to the original equation in [Mahadevan]. Compared to (1), the linear model of (2) explicitly includes the power contribution of individual ports. However, the model assumes that a port consumes power only when there is traffic flowing through it. In absence of traffic, the power contribution of a port is null, whether the port is enabled or disabled. As we show later on, an idle port does consume power in relevant amounts when enabled. By neglecting this tangible contribution to power consumption, the model of (2) does not capture the energy-saving benefits of selectively disabling network ports (and associated links) on the basis of network-wide FTRA optimizations.

The methodology used in [Mahadevan] for collecting power measurements in presence of traffic is very similar to ours. For cost reasons, the number of traffic generators available for the experiments is typically much smaller than the number of ports in a switch or router. However, in the case of an Ethernet switch all ports can still be fully loaded by injection of broadcast traffic at only one of the ports. This is achieved by attaching one port to a traffic generator, another port to a traffic sink, and then looping back all the remaining port pairs. If the spanning tree protocol (STP), originally specified in the IEEE 802.1D standard, is enabled for the switch, every port operates at full capacity in one direction only, as either a receiver or a transmitter. If STP is disabled, all ports operate at full capacity in both directions, since the switch is internally flooded by the full replication of frames that occurs at every port.

Considering the similarities with our work, it is worth summarizing the key results collected in [Mahadevan]: (1) The overall power consumption is sensitive to the number of ports that handle traffic. (2) The power contribution of a single port grows with the configured port capacity. (3) The overall power consumption shows little sensitivity to traffic load variations and no sensitivity at all to the packet size distribution.

The authors conclude that system vendors have plenty of opportunities to save energy by adopting rate adaptation techniques at all timescales.

3.1.3 Tamm et al., 2010

In [Tamm], Tamm et al. deliver a comprehensive study of the distribution of power consumption among the functional components of a large set of Alcatel-Lucent network systems, including optical switches, Ethernet switches, and IP routers. The results help identify key hotspots for energy savings in systems designs, and offer high-level indications of the benefits that may derive from the introduction of BTRA/PTRA-capable hardware in circuit packs. All power measures are obtained from typical ratings of common hardware components and not directly from experimental measurements. Also, the power distribution models presented do not provide a mapping of system and traffic configurations onto power consumption levels, and therefore do not supply the information needed for the operation of FTRA techniques.

3.1.4 ECR Initiative, 2010

The Energy Consumption Rating (ECR) Initiative [ECR] is the ongoing effort of a (small) industry consortium to establish a better metric for qualifying the energy efficiency of network systems than the de facto standard given by the ratio between power rating (maximum power consumption) and nominal maximum throughput. The scalar indices defined by the initiative, which are computed as ratios of weighted sums of power and throughput measures for a fixed set of load levels, do provide a more accurate indication of the energy efficiency of a system and of the energy savings that could be achieved with the introduction of BTRA/PTRA capabilities. However, such indices also maintain a monolithic view of the energy performance of the system, which makes them unsuited for supporting FTRA techniques.

3.2 Reference Model for Energy Profiling

We define our model for energy profiling with two primary goals. First, we want to create a system description that FTRA techniques can immediately utilize in their network-wide energy-optimization procedures. Second, we want to obtain a clear measure of the margin of improvement for energy efficiency that exists in current equipment, and therefore of the energy savings that could be enabled by the introduction of PTRA/BTRA capabilities in data-path hardware components.

Equation (3) formally defines our energy profiling model:

$$P_S = P_C + \sum_{i=1}^{N_L} P_{L,i} + \sum_{j=1}^{N_P} \left(P_{0,j} + P_{E,j} + T_{IN,j} \cdot \rho_{IN,j} + T_{OUT,j} \cdot \rho_{OUT,j} \right), \tag{3}$$

where $P_{0,j}$ is the power consumed by a port j when loaded with an SFP and disabled (shortly, the *base power* of port j), $P_{E,j}$ is the additional power consumed by port j in its current configuration when loaded with the SFP of $P_{0,j}$, enabled for operation, and connected by a link to a peering interface (shortly, the *enabled power* of port j), $T_{IN,j}$ is the additional power consumption of the port when, being loaded and connected, it receives traffic at its full input capacity (shortly, the *input saturation power* of port j), and $T_{OUT,j}$ is the additional power consumption of the port when, being loaded and connected, it transmits traffic at its full output capacity (shortly, the *output saturation power* of port j). The two variables $\rho_{IN,j}$ and $\rho_{OUT,j}$ represent the current utilization of port j in the input and output directions $(0 \le \rho_{IN,j} \le 1,$ $0 \le \rho_{OUT,j} \le 1)$.

Compared to the model of Equation (2), our model splits the power contribution of each port into four components, of which only two are traffic-dependent. The isolation of the base component $P_{0,j}$ shows the benefit of unplugging the SFP out of a port that is never used. The isolation of the enabled component $P_{E,j}$ of the port contribution is essential to the application of the model to the optimization procedures of FTRA techniques. Without the enabled power it would be impossible to appreciate the energy-saving benefits of disabling a port that temporarily does not handle traffic. In the model of Equation (2), the power consumption of a port that is enabled and idle is null, just like the power consumption of a disabled port. The distinction between input and output load is also new in Equation (3), although the constraints that we face in the design of the power measurement experiments prevent us from bringing it to fruition, as we discuss later on. The two variable components of the port power consumption define the distance of the system from the ideal energy-follows-load behavior: a marginal excursion of power measures from minimum to maximum load, as compared to the offset component of the port, indicates that the introduction of BTRA/PTRA capabilities could yield important energy savings.

It is important to remark that the base power $P_{0,j}$, the enabled power $P_{E,j}$, and the load-saturation terms $T_{IN,j}$ and $T_{OUT,j}$ are not fixed for a port, but depend instead on the specific configuration in which port j is found. The base power $P_{0,j}$ obviously depends on the type of SFP that is plugged into the port. For ports that do not work with SFP's (SUT2 and SUT3 have integrated 10/100BASE-TX ports with RJ-45 connectors) the based power $P_{0,j}$ is null. Configuration parameters that define the value of the enabled power $P_{E,j}$ include the type of SFP installed (for ports that work with SFP's) and the operating capacity (e.g., $10\,Mbps$, $100\,Mbps$, or $1\,Gbps$ in the case of Gigabit Ethernet (GbE) ports). The saturation load parameters can be sensitive to the packet size distribution and to the amount of processing required by each packet (the latter is true for the ports of a router much more than for the ports of an Ethernet switch).

4. Experimental Testbed

This section provides an overview of the equipment used in the experiments and of the constraints that the equipment imposes on their execution. We start with a list of definitions and conventions that we follow in the presentation of the results.

4.1 Definitions and Units

- Each experiment is labeled $x.y$, where the value of x designates the SUT ($x = S$ for SUT1, $x = F$ for SUT2, and $x = R$ for SUT3) and y is the unique numeric identifier of the experiment.
- Traffic rates are measured in multiples of bits per second $[bps]$: $1\,Gbps = 10^9\ bps$, $1\,Mbps = 10^6\ bps$, and $1\,kbps = 10^3\ bps$.
- Power consumption is measured in watts $[W]$.
- We refer to an SUT port as *loaded* if it has a small form-factor pluggable optics module attached (SFP or XFP), and as *empty* otherwise.
- We refer to a loaded SUT port as a *connected* port if a network cable connects the port to a peering interface on the same system or on a traffic generator/sink, and as *disconnected* otherwise.
- We refer to an SUT data port as *enabled* if it is configured for operation through the SUT management interface, and as *disabled* otherwise. In general, a port can be enabled and disabled when it is empty, loaded but disconnected, and connected. We are mostly interested in the distinction between the enabled and disabled states in the particular case where the port is connected, because this is the kind of state transitions that is controlled by FTRA techniques.
- We refer to a connected SUT port as *input-busy* if it receives traffic from the attached network cable and as *input-idle* otherwise.
- We refer to a connected SUT port as *output-busy* if it receives traffic from the switching fabric of the system and as *output-idle* otherwise.
- We refer to an input-busy (output-busy) SUT port as *input-saturated* (*output-saturated*) if it receives traffic in proximity of its capacity.
- We refer to an input-busy (output-busy) SUT port as *input-overflowing* (*output-overflowing*) if it receives traffic in excess of its capacity.

4.2 Laboratory Equipment

The laboratory testbed used for execution of the power measurement experiments includes the following items (a distinct icon is used for identification of each item in the drawings that describe the experimental setups).

- Systems under test:

  One SUT1 instance

  One SUT2 instance

  One SUT3 instance.

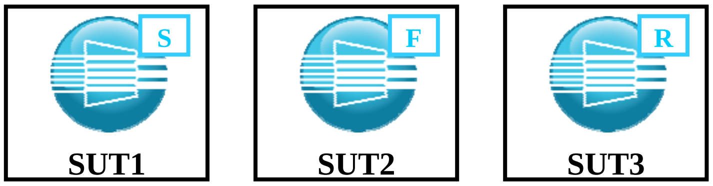

Figure 1. SUT icons.

The following SFP modules are available for connecting cables to the SUT ports:

One (1) 10GBASE-LW/LR XFP module (model number 3HE00564AA) for the 10 *Gbps* Ethernet ports on SUT1.

One (1) 10GBASE-LW/LR XFP module (model number 3HE00564CA) for the 10 *Gbps* Ethernet ports on SUT1.

One (1) 1000BASE-LX SFP module (model number 3HE00028AA) for 10/100/1000 Ethernet ports on all SUT's.

One (1) 1000BASE-LX SFP module (model number 3HE00867CA) for 10/100/1000 Ethernet ports on all SUT's.

Two (2) 1000BASE-SX SFP modules (model number 3HE00027AA) for 10/100/1000 Ethernet ports on all SUT's.

Six (6) 1000BASE-TX SFP Copper modules (model number: 3HE00062AA) with RJ-45 connector for Cat5 copper cable.

Twenty-four (24) 1000BASE-TX SFP Copper modules (model number: GLC-T-MD) with RJ-45 connector for Cat5 copper cable.

- Traffic generators:

Two (2) Gateway Desktop SX2801-07e PC's with 3.0 *GHz* Intel® Pentium® Dual-Core processor E5700, 6 *GB* memory, 1 *TB* hard drive, 10/100/1000 Ethernet LAN interface (1000BASE-TX RJ-45 port), and Linux OS (Ubuntu Release 10.10). The *iperf* utility is used for configuration and operation of the traffic sources. Each traffic generator PC can both transmit and receive traffic to and from the SUT. In the drawings that describe specific experiments, the number of data links between the SUT and the PC traffic generator icon indicate how many PC's are used. The direction of the arrow points on each link indicates if the corresponding PC traffic generator is used as a traffic source only, as a traffic sink only, or as both. The absence of arrow points indicates that the cables are connected both no traffic flows through them.

One (1) Spirent SmartBit SMB-200 chassis (firmware version 6.7, umbrella SmartBit release 10.51) with two (2) SmartMetrics 10/100BASE-TX Ethernet SmartCards (RJ-45 copper interface) and two (2) GX-1405B 1000BASE-SX Ethernet SmartCards (optical interface). In the drawings that describe specific experiments, the number of data links between the SUT and the SMB traffic generator icon indicate how many Ethernet ports are used for each of the two types available. The direction of the arrow points on each link indicates if the corresponding SMB traffic generator port is used as a traffic source only, as a traffic sink only, or as both. The absence of arrow points indicates that a cable is connected but no traffic flows through it.

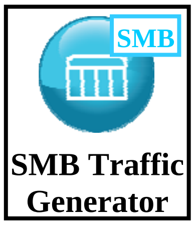

Figure 2. Traffic generator icons.

- DC power supply:

  Xantrex Technology XKW 1kW programmable DC power supply. Form factor: 1.75 inches (1U) in 19-inch rack package. Output ratings: $0-60\,V$ voltage, $0-18\,A$ current, $1080\,W$ power.

Figure 3. DC power supply icon.

- Power meter:

  Extech Instruments 380801 true RMS single-phase power analyzer. Power range: 200/2000W. Power resolution: 0.1/1W. Voltage range: 200/750V. Voltage resolution: 0.1/1V. Current range: 2/20A. Frequency range: 40Hz to 20kHz. Frequency resolution: 1Hz to 10Hz.

- PC for data acquisition from power meter and for configuration and monitoring of the SMB traffic generator:

  Dell Latitude 630 laptop (Windows XP OS) with the following software installed:

  Extech Instrument application for Windows XP for data logging from the power meter. Connected to power meter via serial port.

  Spirent SmartWindows/SmartApplication/SmartLibrary version 9.5 for SMB configuration and monitoring. Connected to the SMB management interface via serial port.

  We refer to the combination of the power meter and data-acquisition PC as the *power meter station*.

Figure 4. Power meter station icon.

- PC for configuration and monitoring of the SUT:

  Gateway desktop (Ubuntu 10.10) connected to the SUT management interface via serial port.

4.3 *Testbed Setups*

The testbed setups are slightly different for SUT1 versus SUT2 and SUT3 because the switch works with an ordinary 120V AC power supply and the routers rely on a DC power supply.

4.3.1 SUT1 Reference Testbed Setup

SUT1 works with a $120\,V,\ 60\,Hz$ AC power supply, which is accessible directly at the AC outlet of the power meter. Figure 5 shows the reference testbed setup for SUT1. The power measurements apply to the AC power supply of SUT1.

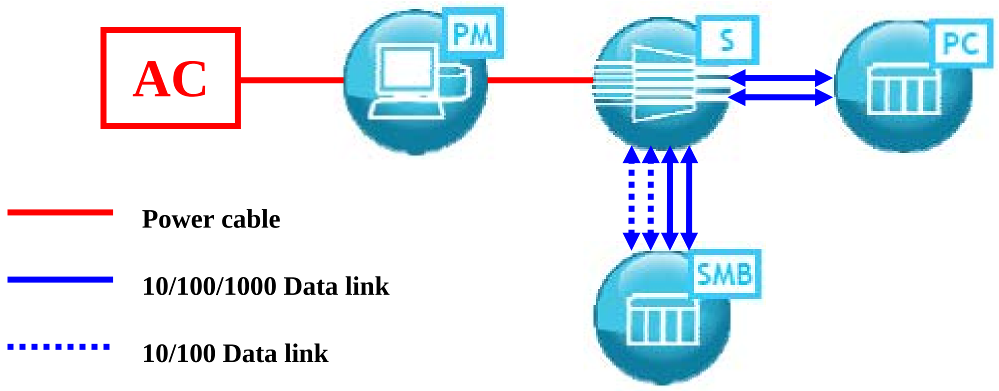

Figure 5. Reference testbed setup for SUT1.

4.3.2 SUT2 and SUT3 Reference Testbed Setup

SUT2 and SUT3 work with a $-48\,V$ DC power supply that is obtained from the Xantrex Technology XKW 1kW module. In our testbed we use one power supply feed, without a backup feed. Figure 6 shows the reference testbed setup for SUT2 (the testbed setup is identical for SUT3, with the obvious exception of the SUT icon).

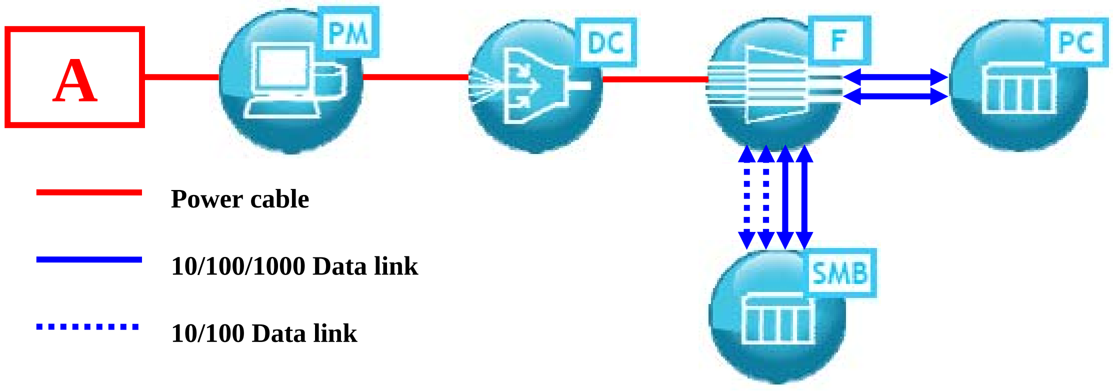

Figure 6. Reference testbed setup for SUT2.

The power meter provides power consumption measurements for the AC power supply of the DC power supply module. The power meter readings include the internal power dissipation of the DC power supply module. Since different DC power supply modules contribute different power dissipation levels under identical DC power draws, and more specifically because we expect the DC power supply module to operate in a region of the efficiency curve where the efficiency is far from maximum (less than $100\,W$ output power, versus a maximum available of $1080\,W$), we decide to report measurements that are based on estimates of the power contributions of the SUT, and not just on the values read at the power meter. We describe the measurement calibration process in the following section.

5. *Power Measurements Methodology*

The goal of our power measurement experiments is to compute for each system the parameters of the linear model of (3), namely the idle chassis power P_C, the idle line card power $P_{L,i}$ for every line card i, the base power $P_{0,j}$ for every port j that is loaded with an SFP, and the enabled power $P_{E,j}$, input saturation power $T_{IN,j}$, and output saturation power $T_{OUT,j}$ for every port j that is connected and enabled.

In this section we provide details about the methodology that we use to compute the parameters, with particular emphasis on special arrangements that we make to address constraints and limitations of our testing equipment.

5.1 *Power Consumption Estimation for SUT2 and SUT3*

Both SUT2 and SUT3 work with a DC power supply ($-48\,V$, $-60\,V$, or $+24\,V$ DC). The option of a redundant power supply feed is available with all voltage levels. In our experiments, a single-feed $-48\,V$ DC power supply is obtained from a Xantrex Technology XKW 1kW programmable DC power supply module (shortly the *DC module*).

The use of an external, third-party component for powering SUT2 and SUT3 raises a fundamental question about the nature of their energy profiles: should the profiles include or exclude the power dissipation that occurs in the DC module?

We conclude that the power dissipation of the DC module should be excluded, so that only the intrinsic properties of the SUT are retained in the energy profile. As the type of the DC module changes, knowledge of the efficiency curve of the DC module used in each occasion enables the immediate conversion of the intrinsic parameters of the SUT into combined values that incorporate the characteristics of the DC module.

When we measure the power consumption of SUT2 or SUT3 with the AC power meter, the meter reading P_{AC} combines the power dissipation of the SUT and of the DC module $(P_{AC} = P_{SUT} + P_{DC}).$ Our DC module contains a voltmeter and an ammeter that provide readings for the output voltage V_{DC}^{Out} (resolution $\Delta V_{DC}^{Out} = 0.1V$) and the output current I_{DC}^{Out} (resolution $\Delta I_{DC}^{Out} = 0.1\,A$). We could think of using the readings on the two instruments embedded in the DC module to measure the power dissipation in the SUT. The decision depends on the resolution errors of the power measures obtained by combining the readings on the DC module instruments and of those provided by the power meter.

To compare the resolution errors of the DC module and AC power meter (where the resolution of the power reading is $\Delta P_{AC} = 0.1W$), we consider an example where the actual power consumption in the SUT is $P_{SUT} = 50\,W.$ In this case, the relative resolution error with the AC power meter is $\varepsilon_{AC} = \pm 0.05/50 = \pm 0.1\%.$ With the voltmeter and ammeter of the programmable DC power supply, the relative resolution error would be $\varepsilon_{DC} = \pm[(0.05/48) + (0.05/1.04)] = \pm 4.9\%,$ which is about 50 times larger than the relative resolution error of the AC power meter.

To proceed with the estimation of the power dissipated by the SUT we assume that the power dissipation in the DC module is the sum of a fixed term $P_{DC}^{(0)}$ and a variable term that depends on the power dissipated by the SUT:

$$P_{DC} = P_{DC}^{(0)} + \alpha(P_{AC}) \cdot P_{SUT}\,,$$

We refer to the constant term $P_{DC}^{(0)}$ as the *base power* of the DC module, and to the function $\alpha(P_{AC})$ as the *inefficiency* of the DC module.

We can now express the power meter reading as a function of the power dissipated by the SUT:

$$P_{AC} = P_{DC}^{(0)} + [1 + \alpha(P_{AC})]\, P_{SUT}\,.$$

We measure $P_{DC}^{(0)}$ as the reading on the power meter when the SUT is powered off:

$$P_{DC}^{(0)} = 38.9W.$$

To estimate the inefficiency $\alpha(P_{AC})$, we must rely on the readings of the output voltage and current $(V_{DC}^{Out}, I_{DC}^{Out})$ offered by the DC module. We collect voltmeter and ammeter readings under a series of different operating conditions for the SUT, which correspond to different values of P_{AC}. The results of the experiments are reported in Table 1 below (the reading of the output voltage of the programmable DC power supply is always $V_{DC}^{Out} = 47.8\,V$). The samples α_i of the function $\alpha(P_{AC})$ are computed as follows:

$$\alpha_i = (P_{AC}^{(i)} - 38.9)/(47.8 \cdot I_{DC}^{Out(i)}) - 1.$$

AC power P_{AC} [W]	**DC current I_{DC}^{Out} [A]**	α samples
130.6	1.4	0.370
132.3	1.4	0.396
135.9	1.5	0.353
132.8	1.4	0.403
131.5	1.4	0.384
131.8	1.4	0.388
136.1	1.5	0.356
152.3	1.8	0.318
151.3	1.7	0.383
147.6	1.7	0.338
148.6	1.7	0.350
153.0	1.8	0.326

Table 1. Measurements used for the estimation of the linear coefficient α.

We use the GNU Octave language [Octave] to interpolate the values in Table 1 with polynomials of degree ranging from 0 to 5. The plots of the polynomials are shown in Figure 7. We expect the actual profile of the function $\alpha(P_{AC})$ to be monotonic (the efficiency of the DC module should improve as the input power increases) and convex ($\alpha(P_{AC})$ should converge to an asymptotic minimum value as P_{AC} keeps growing). We choose the polynomial of degree 2 to represent the function $\alpha(P_{AC})$ because it is the only one, out of all polynomials plotted in Figure 7, that consistently exhibits both expected behaviors. The polynomial is defined as follows:

$$\alpha(P_{AC}) = 2.281 - 0.02505\, P_{AC} + 8.08 \cdot 10^{-5}\, {P_{AC}}^2 .$$

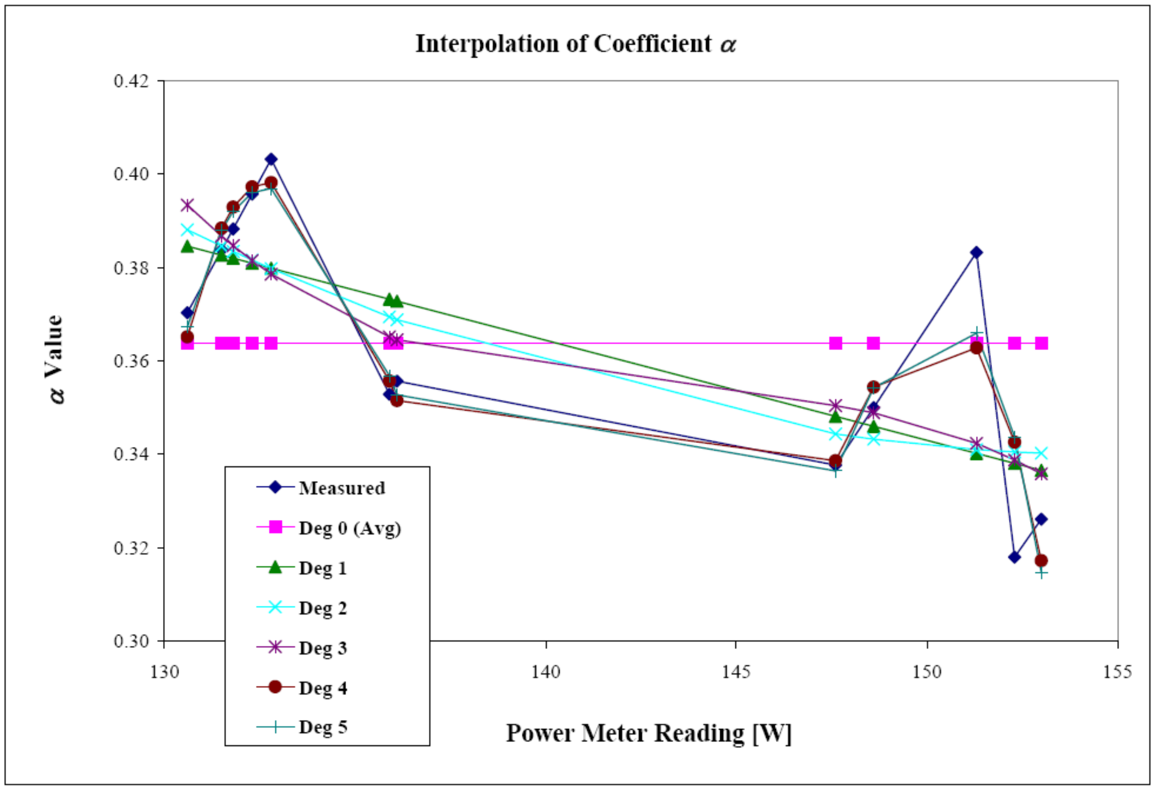

Figure 7. Interpolation of the α samples with polynomials of degree 0 through 5.

For a given reading P_{AC} of the power meter, we obtain the estimated power consumption of the SUT as follows:

$$P_{SUT} = \frac{P_{AC} - P_{DC}^{(0)}}{1 + \alpha(P_{AC})}.$$

5.2 *Estimation of Port Parameters*

Most of our power measurement experiments focus on the quantification of port parameters. In fact, we must compute the base power $P_{0,j}$, the enabled power $P_{E,j}$, the input saturation power $T_{IN,j}$, and the output saturation power $T_{OUT,j}$ for every port j, where j designates not just a physical port, but also a specific choice of SFP (BASE-TX, BASE-SX, or BASE-LX) and configured capacity ($10\ Mbps$, $100\ Mbps$, or $1\ Gbps$). As a result, for each type of 10/100/1000 port we have to compute three values for the base power (one for each type of SFP), nine values for the enabled power (one per port rate per type of SFP), and likewise for the input and output saturation power.

In the estimation of a parameter, we configure the testbed so that the highest number of ports possible operates in the conditions that define the parameter. Such “highest number of ports possible” is subject to a double constraint: first, the number of available SFP’s may be smaller than the number of ports (we do have thirty BASE-TX SFP’s, but only two BASE-SX and two BASE-LX SFP’s); second, the number of available traffic sources/sinks is always limited (all we have are two PC’s with one 10/100/1000BASE-TX port each and an SMB with two

10/100BASE-TX ports plus two 1000BASE-SX optical ports). The goal of maximizing the number of ports that operate in the conditions defined by the target parameter is to minimize the effect of measurement errors, in particular those associated with the relatively coarse resolution of the AC power meter $(\pm 0.05W)$.

For base power calculations, we simply connect and enable/disable as many ports as we can load with SFP's of the target type. For saturation power calculations that involve the presence of packet traffic, when the number of SFP's available is higher than two (in the BASE-TX case only) we use network cables to loop back the extra ports and feed the input port with broadcast traffic. With spanning-tree protocol (STP) enabled, half of the loop-back ports transmit network traffic and the other half receive network traffic. With STP disabled, every loop-back port works simultaneously in transmission and reception of network traffic.

Having no traffic generators that work with 1000BASE-LX or 10GBASE-LW/LR connectors, we resort to mixed configurations for calculation of the parameters of those types of interfaces: we load the system at a port of a different type whose parameters are already known (e.g., a 1000BASE-TX port), and then use one cable of the target type to loop-back two identical interfaces, making sure that traffic flows through the cable at the desired rate.

Special loop-back SFP's that return output traffic back to the input of the same port could also be used in the experiments to increase the port count and therefore minimize the effects of the resolution error of the power meter. However, those SFP's are never deployed in commercial applications and their power consumption is much lower than that of all other types of SFP's (we provide experimental data later on). As a consequence, we avoid using loop-back SFP's in our experiments.

5.3 Service Configuration Constraints in SUT2 and SUT3

The firmware version (2.1) installed in our instances of SUT2 and SUT3 only supports a point-to-point Ethernet service called E-Pipe that has many elements in common with the E-Line service specified by the Metro Ethernet Forum (MEF) [MEF]. The firmware does not allow hosting more than one service per physical port. As a consequence, it is not possible to aggregate traffic from multiple *access ports* (the six 10/100 Ethernet ports on SUT2 and on the adapter card of SUT3) into one *network port (*one of the two 10/100/1000 Ethernet ports on SUT2 and on the adapter card of SUT3). This restriction, which no longer exists in more recent versions of the firmware, makes it impossible to load each of the network ports at a rate higher than 100 *Mbps*.

5.4 Identity of Input and Output Saturation Power Values

In all system configurations that our limited availability of traffic generators and sinks allows us to devise, the total traffic entering the target ports matches the total traffic that exits the same ports. While this is not necessarily the case for individual ports, the statement generally holds for the entire set of ports involved in the experiments. We therefore cannot distinguish between the power consumption contributions of ports that only receive traffic from a network cable and ports that only transmit traffic to a network cable. As a consequence, from now on we approximate the port behavior by assuming that the values of the input saturation and output saturation power are identical: $T_{IN,j} = T_{OUT,j} = T_j$. We can thus rewrite Equation (3) as follows:

$$P_S = P_C + \sum_{i=1}^{N_L} P_{L,i} + \sum_{j=1}^{N_P}\left[P_{0,j} + P_{E,j} + T_j \cdot \left(\rho_{IN,j} + \rho_{OUT,j}\right)\right]. \qquad (4)$$

While we generally expect the two values to be different, the results of our measurements will show that the error introduced by our approximation is marginalized by the little sensitivity of the overall power consumption to the presence of traffic at the network ports.

We remark that, according to Equation (4), the presence of traffic at a port contributes maximum power when both $\rho_{IN,j} = 1$ and $\rho_{OUT,j} = 1$, i.e., when the saturation power T_j is multiplied by a factor 2, not 1.

6. *Experimental Results*

In this section we present the results of our power-measurement experiments, divided into separate per-SUT subsections. We open each subsection with plots that summarize the results for the corresponding SUT, we continue with a tabular presentation of the same data, and conclude with a detailed description of the experiments. The tables with port data are defined per type of SFP. For each value in a table, the x.y label in parentheses identifies the experiment that produced the value.

Unless otherwise noted, the traffic generators attached to the SUT's supply constant-bit-rate UDP traffic encapsulated in $1500\ B$ Ethernet frames.

In the tables with per-port parameters, the *One Port* column contains the values of the port parameters of the linear module, the *N Ports* column (with *N* = 2, 24, 6, 36, or 12 depending on the system and on the type of port) contains the contribution of the parameter projected over the largest set of ports in the system that can take its value (notice that in the case of the saturation power, the value of the parameter is multiplied first by 2 and then by the number *N* of applicable ports), the *System Total* column contains the estimated total power consumption of the system under the projection of the previous column, and the *N Ports Weight* column provides the ratio between the values in the *N Ports* and *System Total* columns.

6.1 Energy Profile of SUT1

6.1.1 Summary of SUT1 Results

In this section we provide visual indication of the way power consumption is distributed over the port states that can be controlled by rate adaptation techniques (Figure 8) and of the maximum impact that per-port power consumption can have at the system level (Figure 9). In Figure 9, the contribution of a single port is multiplied by 24 in the case of 1GbE ports and by 2 in the case of 10GbE ports. The data plotted in Figure 8 and Figure 9 are collected in Table 2 through Table 7.

We stress that Figure 8 shows a power breakdown per individual port, whereas Figure 9 shows the total system power when all ports of a given type are considered. The LW/LR(10G) column is the lowest in Figure 9 despite being the highest in Figure 8 because there are only 2 ports that work with pluggable modules of this type in SUT1, as opposed to 24 ports for all other SFP types.

Figure 8 shows that the enabled power $P_{E,j}$ is the dominant component for all types of ports, contributing more than 50% of the total port power in all cases. Conversely, the traffic-dependent saturation power T_j is always the smallest contributor to the power consumption of a port. We explain this latter result as the combined effect of the monolithic nature of SUT1 and of the lack of PTRA capabilities in the system. The monolithic architecture implies that the central switch fabric must be powered on independently of the number of ports that handle traffic. Then, because it lacks PTRA capabilities, the fabric is unable to adjust its power consumption to the amount of traffic that flows through the individual ports of the system.

Figure 8. (deleted, proprietary)

Figure 9. (deleted, proprietary)

6.1.2 Idle Chassis Power

Table 2. (deleted, proprietary)

6.1.3 Idle Line Card Power

Table 3. (deleted, proprietary).

(1) SUT1 does not have modular components.

6.1.4 Port Power, 1GbE Port with 1000BASE-TX SFP

Table 4. (deleted, proprietary)

(2) The estimated value of $T_{TX}(100\,Mbps)$ is substantially smaller than that of $T_{TX}(10\,Mbps)$, with the result that with configured port rate of 100 $Mbps$ the projected total system power consumption is lower than with configured rate of 10 $Mbps$. We would reasonably expect the total system power not to be lower at 100 $Mbps$ than it is at 10 $Mbps$. We explain this counter-intuitive outcome with the relatively large inaccuracy of the power meter with respect to the power increments that are induced by the presence of traffic and that constitute the object of the power measurements in this experiment.

6.1.5 Port Power, 1GbE Port with 1000BASE-SX SFP

Table 5. (deleted, proprietary)

(3) The 1000BASE-SX SFP's could only exchange traffic with the SMB traffic generator (no loop-back cable).Those ports cannot be configured for operation at rates lower than 1 $Gbps$.

6.1.6 Port Power, 1GbE Port with 1000BASE-LX SFP

Table 6. (deleted, proprietary)

(4) No optics cable compatible with the 1000BASE-LX SFP's was available. The enabled power measure was obtained with disconnected ports.

6.1.7 Port Power, 10GbE Port with 10GBASE-LW/LR XFP

Table 7. (deleted, proprietary)

6.1.8 Description of SUT1 Experiments

This section describes the experiments that produced the SUT1 results listed in the tables of the previous subsections.

6.1.8.1 Experiment S.1

<u>Purpose</u>

Estimate the idle chassis power P_C for SUT1.

<u>Method</u>

Power up SUT1 with all ports empty and disabled and collect power samples for about 10 minutes.

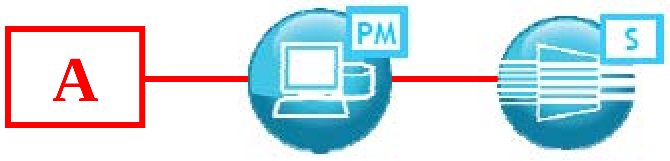

Figure 10. Testbed setup for Experiment S.1.

<u>Results</u>

Deleted, proprietary.

Figure 11. Power measurements with all ports empty and disabled (S.1).

6.1.8.2 Experiment S.2

<u>Purpose</u>

Estimate the base power $P_{0,TX}$, $P_{0,SX}$, $P_{0,LX}$, and $P_{0,LW/LR}$ for the four types of SFP/XFP available.

<u>Method</u>

Power up SUT1 with all ports empty and disabled. Plug in all the SFP's that are available for the target type, so that the effects of the resolution error of the power meter are minimized. Keep all loaded ports disabled. Collect power samples for $600\,s$. Average the power samples. Round the average to the closest digit on the AC power meter.

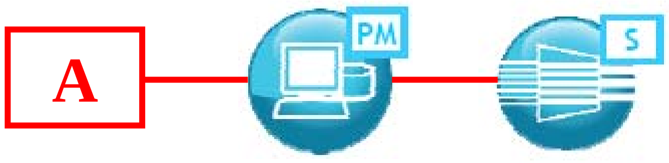

Figure 12. Testbed setup for Experiment S.2.

<u>Results</u>

Deleted, proprietary.

6.1.8.3 EXPERIMENT S.3

<u>Purpose</u>

Estimate the enabled power $P_{E,TX}$, and $P_{E,LW/LR}$ for the 1000BASE-TX SFP and the 10GBASE-LW/LR XFP.

<u>Method</u>

Power up SUT1 with all ports empty and disabled. Plug in all the SFP's that are available for the target type, so that the effects of the resolution error of the power meter are minimized. Connect each loaded port to a respective peering port on the switch using a loop-back network cable. Enable all ports that are connected. Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

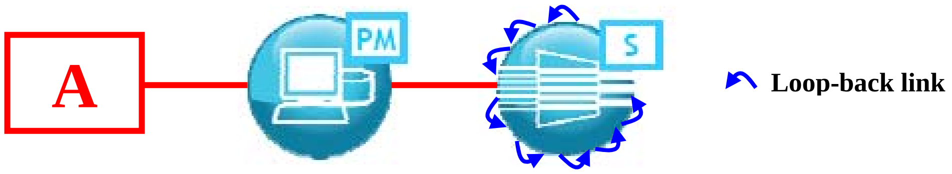

Figure 13. Testbed setup for Experiment S.3

<u>Results</u>

Deleted, proprietary

Figure 14. Linearity of enabled power contributions from ports loaded with 1000BASE-TX SFP's.

6.1.8.4 EXPERIMENT S.4

<u>Purpose</u>

Estimate the enabled power $P_{E,SX}$ for the 1000BASE-SX SFP.

<u>Method</u>

The optics cable for the 1000BASE-SX SFP's has different connectors at the two ends. This makes it impossible to load the idle ports with a loop-back cable. Instead, the ports must be loaded by direct connection to respective ports on the SMB traffic generator. The measurement procedure is as follows:

Power up SUT1 with all ports empty and disabled. Plug in the two available 1000BASE-SX SFP's. Connect each loaded port to a respective peering port on the SMB, keeping the traffic generator idle. Enable the two connected ports. Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

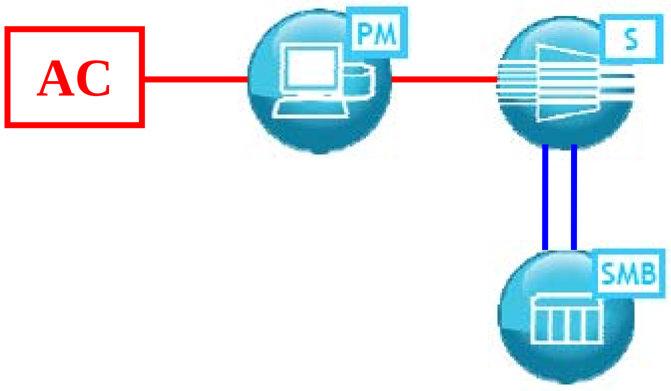

Figure 15. Testbed setup for Experiment S.4.

<u>Results</u>

Deleted, proprietary

6.1.8.5 Experiment S.5

<u>Purpose</u>

Estimate the enabled power $P_{E,LX}$ for the 1000BASE-LX SFP.

<u>Method</u>

No optics cable is available for the 1000BASE-LX SFP. The estimation of the enabled power is executed without cable connections on the two ports loaded with 1000BASE-LX SFP's. The measurement procedure is as follows:

Power up SUT1 with all ports empty and disabled. Plug in the two available 1000BASE-LX SFP's. Enable the two loaded ports. Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

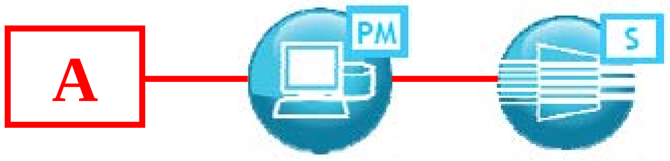

Figure 16. Testbed setup for Experiment S.5.

Results

Deleted, proprietary

6.1.8.6 EXPERIMENT S.6

Purpose

Estimate the saturation power T_{TX} for the 1000BASE-TX SFP.

Method

Power up SUT1 with all ports empty and disabled. Plug in the 24 1000BASE-TX SFP's. Connect two ports to respective peering ports on the PC traffic generators via copper cables with RJ-45 connectors (the PC traffic generators work with RJ-45 generators and can configure their network cards for operation at $10\,Mbps$, $100\,Mbps$, and $1\,Gbps$, whereas the SMB ports only support operation at $1\,Gbps$). Enable all ports on SUT1. Enable the spanning tree protocol. Use one PC for generation of constant-bit-rate broadcast traffic at the operating rate of the network card. Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

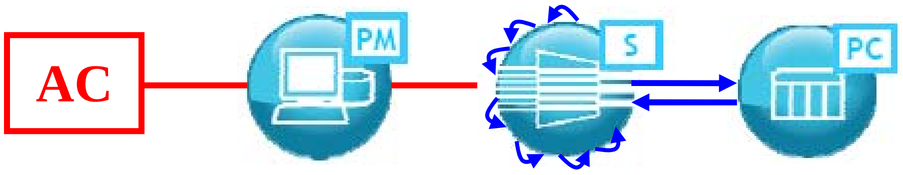

Figure 17. Testbed setup for Experiment S.6.

Results

Deleted, proprietary

6.1.8.7 EXPERIMENT S.7

Purpose

Estimate the saturation power T_{SX} for the 1000BASE-SX SFP.

Method

Power up SUT1 with all ports empty and disabled. Plug the two 1000BASE-SX SFP's into respective ports of SUT1. Connect the two ports to respective peering ports on the SMB traffic generator via asymmetric optics cables. Enable the two connected ports on SUT1. Use one port for generation of $1\,Gbps$ constant-bit-rate traffic destined for the second port. Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

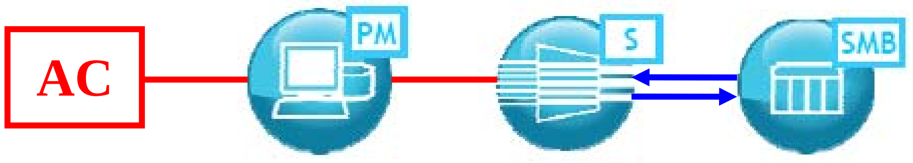

Figure 18. Testbed setup for Experiment S.7.

Results

Deleted, proprietary

6.1.8.8 EXPERIMENT S.8

Purpose

Estimate the saturation power $T_{LW/LR}$ for the 10GBASE-LW/LR XFP.

Method

We do not have a traffic generator with a $10\,Gbps$ network interface. The only way to run a $10\,Gbps$ port at full capacity is to connect it to the second 10GbE port via a loop-back optics cable, and then load the switch with $1\,Gbps$ broadcast traffic from a third (1GbE) port with spanning tree protocol disabled. This way each 10GbE port receives and transmits traffic at full $10\,Gbps$ rate, with plenty of packet losses occurring in the switch.

The measurement procedure is as follows. Power up SUT1 with all ports empty and disabled. Plug the two 1000BASE-LW/LR XFP's into respective 10GbE ports of SUT1. Connect the two 10GbE ports via an optical loop-back cable. Plug a 1000BASE-SX SFP into a 1GbE port of SUT1. Connect the 1GbE port to a respective port on the SMB. Enable the three ports. Generate broadcast traffic at $1\,Gbps$ rate. Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

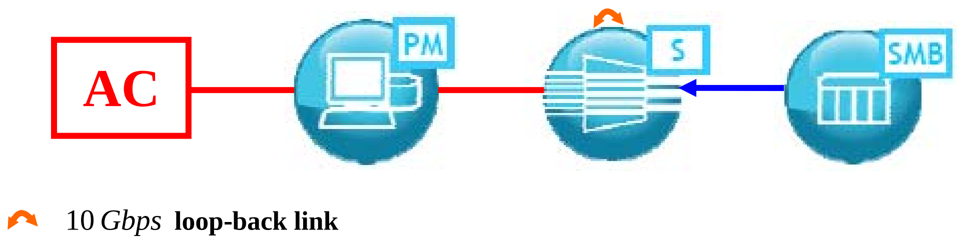

Figure 19. Testbed setup for Experiment S.8.

Results

Deleted, proprietary.

6.2 Energy Profile of SUT2

All the values plotted and listed in this subsection are obtained after subtraction of the estimated DC module contribution from the reading of the AC power meter.

6.2.1 Summary of SUT2 Results

In this section we provide visual indication of the way power consumption is distributed over the port states that can be controlled by rate adaptation techniques (Figure 20) and of the maximum impact that per-port power consumption can have at the system level (Figure 21). In Figure 21, the contribution of a single port is multiplied by 6 in the case of 10/100 Ethernet ports and by 2 in the case of 1GbE ports. The data that produced the plots are collects in Table 8 through Table 13.

Figure 20 shows that the enabled power $P_{E,j}$ is the dominant component for the BASE-TX ports (both 10/100BASE-TX and 1000BASE-TX), but not for the ports with 1000BASE-SX and 1000BASE-TX SFP's, where the base power $P_{0,j}$ gains a larger power consumption share. The traffic-sensitive component T_j, although never dominant, becomes more prominent is a few cases.

Figure 21 emphasizes the difference between the fixed and variable power consumption components in SUT2. The behavior finds the same explanations given for SUT1: lack of PTRA-capable data-path hardware in a monolithic system design. However, the behavior is even more pronounced than in the SUT1 case because of the difference in the number of Ethernet ports (up to 6 ports of the same kind in SUT2, up to 24 ports of the same kind in SUT1).

Figure 20. Deleted, proprietary.

Figure 21. Deleted, proprietary.

6.2.2 Idle Chassis Power

Table 8. Deleted, proprietary.

6.2.3 Idle Line Card Power

Table 9. Deleted, proprietary.

(5) SUT2 does not have modular components.

6.2.4 Port Power, 10/100BASE-TX Ethernet Port

Table 10. Deleted, proprietary.

(6) The base power $P_{0,10/100TX}$ cannot be measured for the 10/100BASE-TX ports of SUT2 because the ports are integrated, without SFP.

6.2.5 Port Power, 1GbE Port with 1000BASE-TX SFP

Table 11. Deleted, proprietary.

6.2.6 Port Power, 1GbE Port with 1000BASE-SX SFP

Table 12. Deleted, proprietary.

(7) The 1000BASE-SX SFP's could only exchange traffic with the SMB traffic generator (no symmetric cable available for loop-back connections). The 1GbE ports of the SMB cannot be configured for operation at rates lower than $1\,Gbps$.

6.2.7 Port Power, 1GbE Port with 1000BASE-LX SFP

Table 13. Deleted, proprietary.

(8) No optics cable compatible with the 1000BASE-LX SFP's was available. The enabled power measure was obtained with disconnected ports.

6.2.8 Description of SUT2 Experiments

This section provides details about the experiments cited in the previous tables.

6.2.8.1 EXPERIMENT F.1

Purpose

Estimate the idle chassis power P_C for SUT2.

Method

Power up SUT2 with all ports empty and disabled and collect power samples for about 10 minutes.

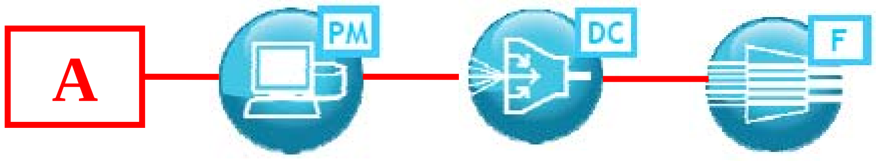

Figure 22. Testbed setup for Experiment F.1.

Results

Deleted, proprietary.l

6.2.8.2 EXPERIMENT F.2

Purpose

Estimate the enabled power $P_{E,\,10/100TX}$ for the 10/100BASE-TX integrated ports.

Method

Power up SUT2 with all ports disabled. Using copper cables connect four 10/100BASE-TX ports to respective RJ-45 ports on the SMB (two ports) and on the two traffic-generator PC's (one port each). Enable the four connected ports. Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

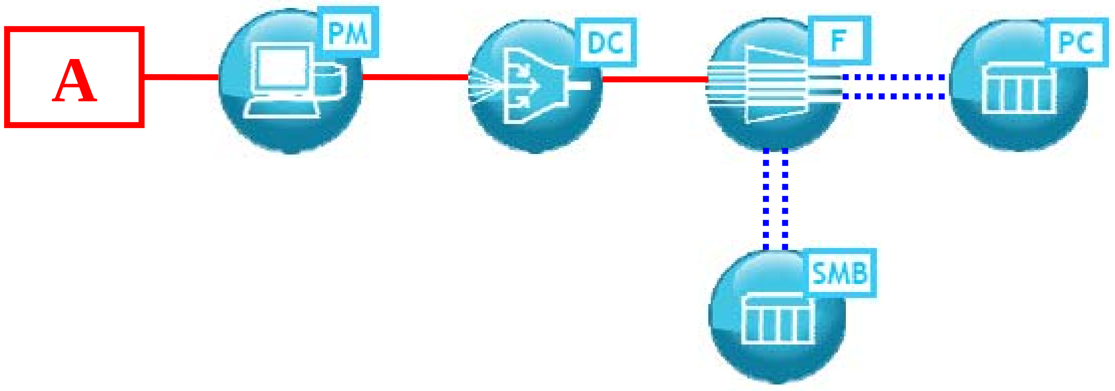

Figure 23. Testbed setup for Experiment F.2.

<u>Results</u>

Deleted, proprietary.

6.2.8.3 EXPERIMENT F.3

Estimate the enabled power $P_{E,\,10/100TX}$ for the 10/100BASE-TX integrated ports.

<u>Method</u>

Power up SUT2 with all ports disabled. Using copper cables connect two 10/100BASE-TX ports to respective RJ-45 ports on the SMB. Enable the two connected ports. Run constant-bit-rate unicast traffic at the full rate configured for SUT2 ports from one port of the SMB to the other. Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

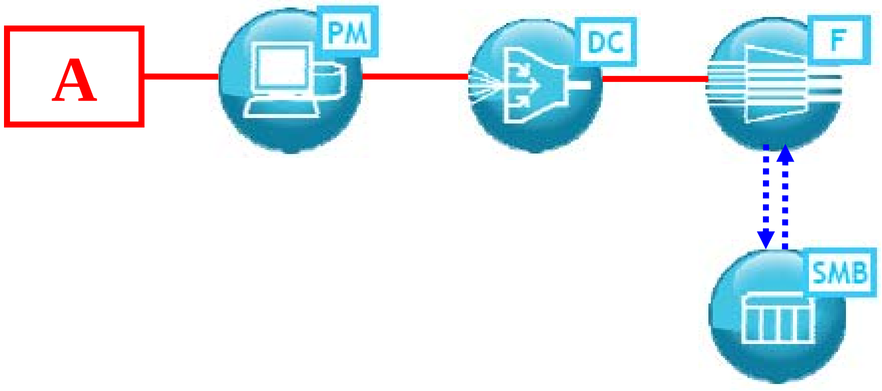

Figure 24. Testbed setup for Experiment F.3.

<u>Results</u>

Deleted, proprietary.

6.2.8.4 EXPERIMENT F.4

<u>Purpose</u>

Estimate the base power $P_{0,TX}$ for a 1GbE port with 1000BASE-TX, 1000BASE-SX, and 1000BASE-LX SFP.

Method

Power up SUT2 with all ports empty and disabled. Load the two 1GbE ports with respective SFP's of the same kind. Keep the loaded ports disabled. Collect power samples for about 10 minutes.

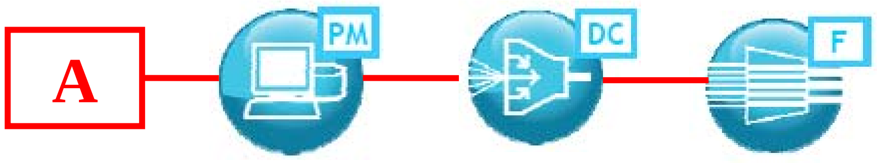

Figure 25. Testbed setup for Experiment F.4.

Results

Deleted, proprietary.

6.2.8.5 EXPERIMENT F.5

Estimate the enabled power $P_{E,TX}(10\ Mbps)$, $P_{E,TX}(100\ Mbps)$, and $P_{E,TX}(1\ Gbps)$ for the 1000BASE-TX SFP's on the 1GbE ports of SUT2. Estimate the enabled power $P_{E,SX}(1\ Gbps)$ for the 1000BASE-SX SFP's on the 1GbE ports of SUT2. Estimate the enabled power $P_{E,LX}(1\ Gbps)$ for the 1000BASE-LX SFP's on the 1GbE ports of SUT2.

Method

Power up SUT2 with all ports disabled. Plug two SFP's of the target type into the two 1GbE ports of SUT2. Enable the ports for operation at the desired rate ($10\ Mbps, 100\ Mbps$, and $1\ Gbps$, based on availability with the target SFP type). Depending on the type of SFP and operating rate under test, connect the enabled SUT2 ports as follows: (a) connect the 1000BASE-TX SFP's to respective RJ-45 ports on the SMB for operation at $10\ Mbps$ and $100\ Mbps$ ($10\ Mbps, 100\ Mbps$, and $1\ Gbps$, based on availability with the target SFP type); (b) connect the 1000BASE-TX SFP's to respective ports on the traffic-generator PC's for operation at $1\ Gbps$; (c) connect the 1000BASE-SX SFP's to respective 1GbE ports on the SMB; or (d) leave the 1000BASE-LX SFP's disconnected. Collect power samples for $600\ s$ after expiration of a $50\ s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

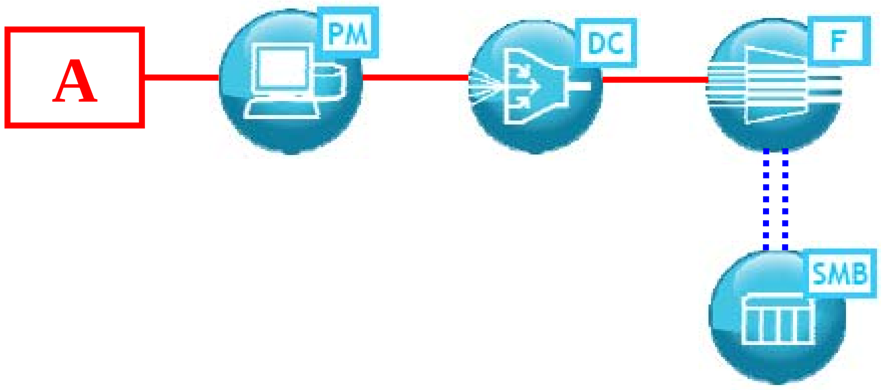

Figure 26. Testbed setup for Experiment F.5 (1000BASE-TX SFP, 10Mbps and 100Mbps).

<u>Results</u>

Deleted, proprietary.

6.2.8.6 EXPERIMENT F.6

<u>Purpose</u>

Estimate the saturation power $T_{TX}(10\ Mbps)$, $T_{TX}(100\ Mbps)$, and $T_{TX}(1\,Gbps)$ for the 1000BASE-TX SFP and the saturation power $T_{SX}(1\,Gbps)$ for the 1000BASE-SX SFP.

<u>Method</u>

Power up SUT2 with all ports empty and disabled. Plug two SFP's of the same kind into respective 1GbE ports of SUT2. Enable the ports for operation at the desired rate ($10\,Mbps, 100\ Mbps$, and $1\,Gbps$, based on availability with the target SFP type). Depending on the type of SFP and operating rate under test ($10\,Mbps, 100\ Mbps$, or $1\,Gbps$, based on availability with the target SFP type), connect the enabled SUT2 ports as follows: (a) connect the 1000BASE-TX SFP's to respective RJ-45 ports on the SMB for operation at $10\ Mbps$; (b) connect the 1000BASE-TX SFP's to respective ports on the traffic-generator PC's for operation at $1\,Gbps$; and (c) connect the 1000BASE-SX SFP's to respective 1GbE ports on the SMB. Use one traffic generator port for full-capacity generation of constant-bit-rate traffic destined for the second port. Collect power samples for $600\,s$ after expiration of a $50\ s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

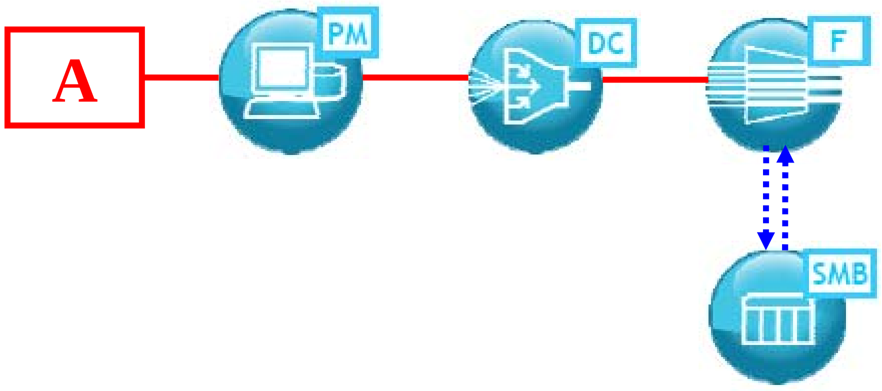

Figure 27. Testbed setup for Experiment F.6 (1000BASE-TX SFP, 10Mbps and 1000Mbps).

Results

Deleted, proprietary.

6.3 Energy Profile of SUT3

The chassis of SUT3 can accommodate up to 8 cards. Two (2) of the slots are dedicated to the *control and switch module* (CSM) cards (one primary, one standby). The other six (6) slots can be populated with adapter cards of different types. Two (2) Ethernet adapter cards, each with 6 x 10/100 Ethernet ports and 2 x 10/100/1000 Ethernet ports, were used in the experiments.

6.3.1 Summary of SUT3 Results

In this section we provide visual indication of the way power consumption is distributed over the port states that can be controlled by rate adaptation techniques (Figure 28) and of the maximum impact that per-port power consumption can have at the system level (Figure 29). In Figure 29, since the system can accommodate up to 6 Ethernet Adapter cards with 6 10/100 Ethernet and 2 1GbE port each, the contribution of a single port is multiplied by 36 in the case of 10/100 Ethernet ports and by 12 in the case of 1GbE ports. Given the modular architecture of SUT3, we must also consider the power contribution of the 6 Ethernet Adapter Cards. The same data that produced the plots of Figure 28 and Figure 29 are listed in Table 14 through Table 19.

Figure 28 confirms a trend already observed for SUT2, where the enabled power $P_{E,j}$ is no longer dominant for all port types and the other power components gain larger shares of the total port power. However, the plots of Figure 29 marginalize the role of the power distribution among per-port components, because the fraction of system power dissipated in the ports is very small compared to the idle power of the chassis and Ethernet Adapter Cards. The conclusion is that the application of FTRA techniques to SUT3 yields only minimal benefits. The margins for PTRA could be higher if it is the case that most of the idle card power is dissipated in data-path hardware components.

The differences between the power contributions of ports and idle cards also stress the uncertainty on the per-port parameters that is induced by the indirect measurement method used for SUT2 and SUT3 in general, and then for the port contributions in presence of traffic.

On one hand, direct measurement of the DC power that feeds SUT2 and SUT3 would reduce the uncertainty caused by the interpolation of the inefficiency function for the DC module. On the other hand, measurements executed with ports that are loaded with traffic exchanged with external peering ports would reduce the error associated with the instrument uncertainty of the AC power meter (in most experiment only a small fraction of the ports available are actually enabled and handling traffic).

Figure 28. Deleted, proprietary.

Figure 29. Deleted, proprietary.

6.3.2 Idle Chassis Power

Table 14. Deleted, proprietary.

6.3.3 Idle Ethernet Line Card Power

Table 15. Deleted, proprietary.

6.3.4 Port Power, 10/100BASE-TX Ethernet Port

Table 16. Deleted, proprietary.

(9) The base power $P_{0,\,10/100TX}$ cannot be measured for the 10/100BASE-TX ports of SUT3 because the ports are integrated in the line card without SFP.

(10) The estimated value of $T_{10/100TX}(100\,Mbps)$ is substantially smaller than that of $T_{10/100TX}(10\,Mbps)$, with the result that with configured port rate of $100\ Mbps$ the projected total system power consumption is lower than with configured rate of $10\ Mbps$. We would reasonably expect the total system power not to be lower at $100\ Mbps$ than it is at $10\ Mbps$. We explain this counter-intuitive outcome with the relatively large inaccuracy of the power meter with respect to the power increments that are induced by the presence of traffic and that constitute the object of the power measurements in this experiment.

6.3.5 Port Power, 1GbE Port with 1000BASE-TX SFP

Table 17. Deleted, proprietary.

6.3.6 Port Power, 1GbE Port with 1000BASE-SX SFP

Table 18. Deleted, proprietary.

(11) The 1000BASE-SX SFP's could only exchange traffic with the SMB traffic generator (no symmetric cable available for loop-back connections). The 1GbE ports of the SMB cannot be configured for operation at rates lower than $1\,Gbps$.

6.3.7 Port Power, 1GbE Port with 1000BASE-LX SFP

Table 19. Deleted, proprietary.

[12] No optics cable compatible with the 1000BASE-LX SFP's was available. The enabled power measure was obtained with disconnected ports.

6.3.8 Description of SUT3 Experiments

This section provides details about the experiments cited in the previous tables.

6.3.8.1 EXPERIMENT R.1

Purpose

Estimate the idle chassis power P_C and the idle Ethernet line card power $P_{L,Ethernet}$ for SUT3.

Method

The SUT3 chassis can be loaded with one or two Control and Switch Module (CSM) instances and one or two 8-port Ethernet Adapter Card (EAC) instances. System configurations with different numbers of CSM and EAC modules always include a FAN module for cooling. In each experiment we power up SUT3 with all ports empty and disabled and collect power samples for about 10 minutes.

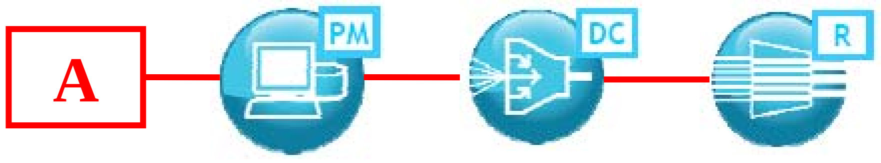

Figure 30. Testbed setup for Experiment R.1.

Results

Table 20. Deleted, proprietary.

6.3.8.2 EXPERIMENT R.2

Purpose

Estimate the enabled power $P_{E,10/100TX}$ for the 10/100BASE-TX integrated ports on SUT3.

Method

Power up SUT3 with all ports disabled. Using copper cables connect four 10/100BASE-TX ports on one of the two EAC modules to respective RJ-45 ports on the SMB (two ports) and on the two traffic-generator PC's (one port each). Enable the four connected ports. Collect power samples for $600s$ after expiration of a $50\ s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

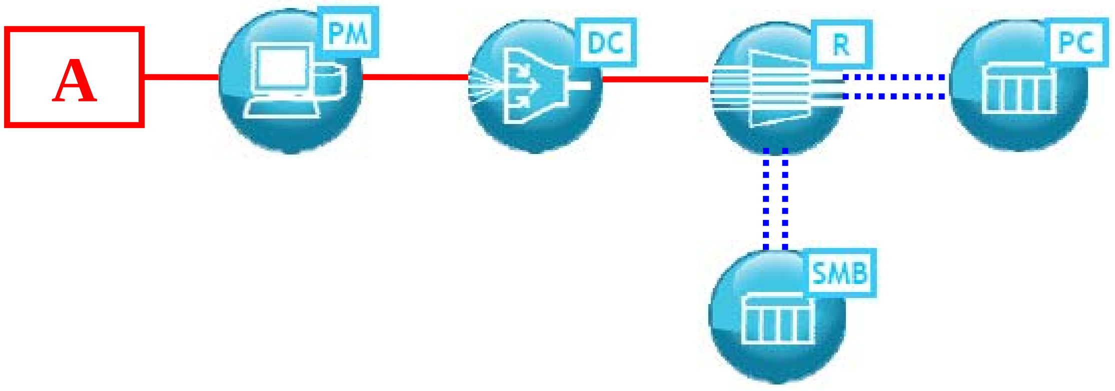

Figure 31. Testbed setup for Experiment R.2.

6.3.8.3 EXPERIMENT R.3

Estimate the enabled power $P_{E,10/100TX}$ for the 10/100BASE-TX integrated ports.

<u>Method</u>

Power up SUT3 with all ports disabled. Using copper cables connect two 10/100BASE-TX ports on the same EAC module to respective RJ-45 ports on the SMB. Enable the two connected ports. Run constant-bit-rate unicast traffic at the full rate configured for the SUT3 ports from one port of the SMB to the other. Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

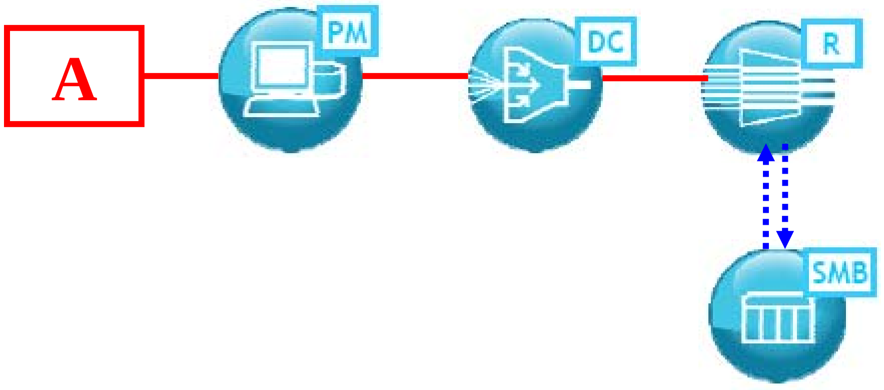

Figure 32. Testbed setup for Experiment R.3.

6.3.8.4 EXPERIMENT R.4

<u>Purpose</u>

Estimate the base power $P_{0,TX}$ for a 1GbE port with 1000BASE-TX, 1000BASE-SX, and 1000BASE-LX SFP.

<u>Method</u>

Power up SUT3 with all ports empty and disabled. With 1000BASE-TX SFP's, load two 1GbE ports in each EAC module. With 1000BASE-SX and 1000BASE-LX SFP's, load one 1GbE port in each EAC module. Keep the loaded ports disabled. Collect power samples for about 600 seconds after expiration of a $50\,s$ warm-up period.

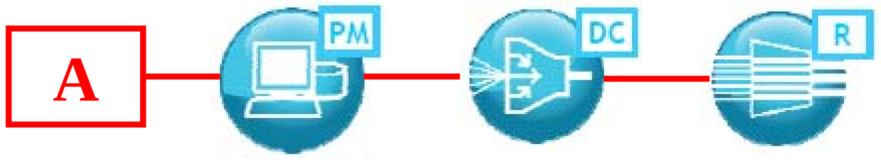

Figure 33. Testbed setup for Experiment R.4.

Results

Deleted, proprietary.

6.3.8.5 EXPERIMENT R.5

Estimate the enabled power $P_{E,TX}(10\,Mbps)$, $P_{E,TX}(100\,Mbps)$, and $P_{E,TX}(1\,Gbps)$ for the 1000BASE-TX SFP's on the 1GbE ports of SUT3. Estimate the enabled power $P_{E,SX}(1\,Gbps)$ for the 1000BASE-SX SFP's on the 1GbE ports of SUT3. Estimate the enabled power $P_{E,LX}(1\,Gbps)$ for the 1000BASE-LX SFP's on the 1GbE ports of SUT3.

Method

Power up SUT3 with all ports disabled. Plug two SFP's of the target type into the two 1GbE ports of SUT3 (4 SFP's for 1000BASE-TX experiments at $10\,Mbps$ and $100\,Mbps$ configured port rates). The ports are on different EAC modules. Enable the ports for operation at the desired rate ($10\,Mbps, 100\,Mbps,$ and $1\,Gbps,$ based on availability with the target SFP type). Depending on the type of SFP and operating rate under test, connect the enabled SUT3 ports as follows: (a) connect the 1000BASE-TX SFP's to respective RJ-45 ports on the SMB and traffic-generator PC's for operation at $10\,Mbps$ and $100\,Mbps$ (see Figure 34); (b) connect two 1000BASE-TX SFP's to respective ports on the traffic-generator PC's for operation at $1\,Gbps$ (Figure 35); (c) connect the 1000BASE-SX SFP's to respective 1GbE ports on the SMB (Figure 36); or (d) leave the 1000BASE-LX SFP's disconnected (Figure 37). Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

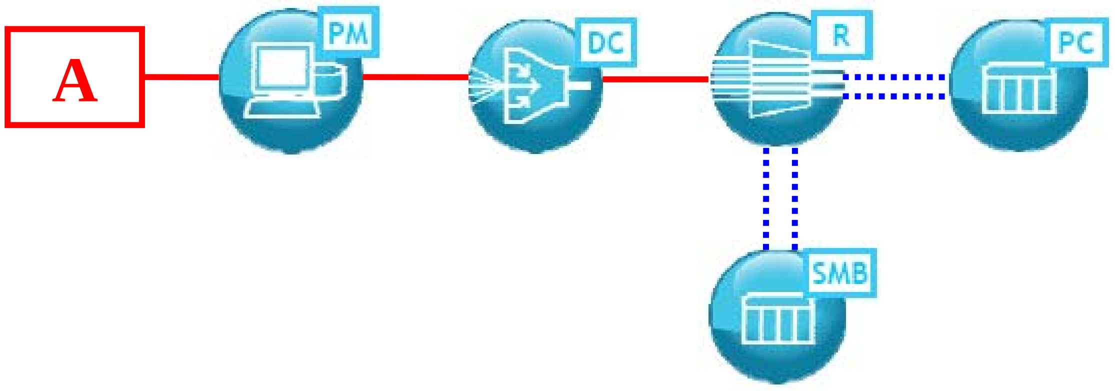

Figure 34. Testbed setup for Experiment R.5 (1000BASE-TX SFP, 10Mbps and 100Mbps).

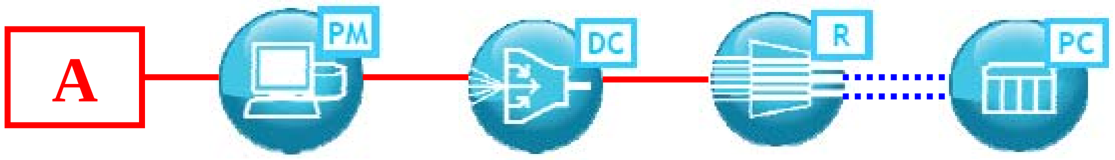

Figure 35. Testbed setup for Experiment R.5 (1000BASE-TX SFP, 1Gbps).

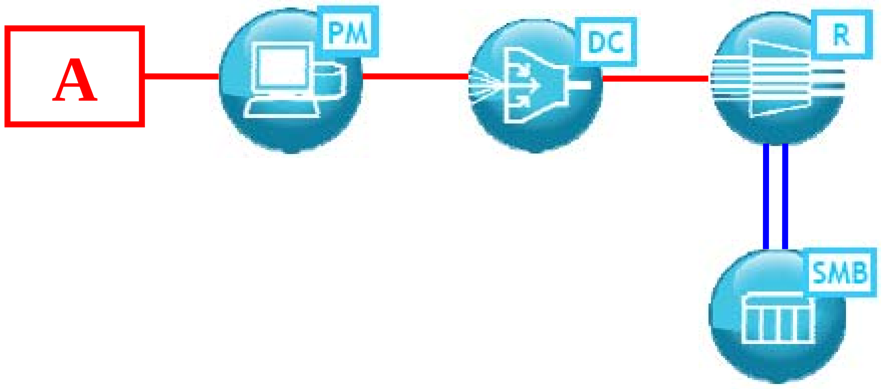

Figure 36. Testbed setup for Experiment R.5 (1000BASE-SX SFP).

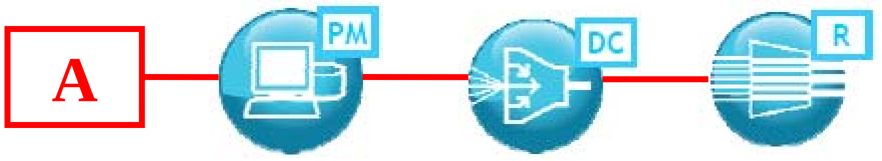

Figure 37. Testbed setup for Experiment R.5 (1000BASE-LX SFP).

<u>Results</u>

Deleted, proprietary.

6.3.8.6 Experiment R.6

Purpose

Estimate the saturation power $T_{TX}(10\,Mbps)$, $T_{TX}(100\,Mbps)$, and $T_{TX}(1\,Gbps)$ for the 1000BASE-TX SFP and the saturation power $T_{SX}(1\,Gbps)$ for the 1000BASE-SX SFP.

Method

Power up SUT3 with all ports empty and disabled. Plug two SFP's of the same kind into respective 1GbE ports of SUT3 on distinct EAC modules. Enable the ports for operation at the desired rate ($10\,Mbps$, $100\,Mbps$, and $1\,Gbps$, based on availability with the target SFP type). Depending on the type of SFP and operating rate under test ($10\,Mbps$, $100\,Mbps$, or $1\,Gbps$, based on availability with the target SFP type), connect the enabled SUT3 ports as follows: (a) connect the 1000BASE-TX SFP's to respective RJ-45 ports on the SMB traffic generator for operation at $10\,Mbps$; (b) connect the 1000BASE-TX SFP's to respective ports on the traffic-generator PC's for operation at $1\,Gbps$; and (c) connect the 1000BASE-SX SFP's to respective 1GbE ports on the SMB. Use one traffic generator port for full-capacity generation of constant-bit-rate traffic destined for the second port. Collect power samples for $600\,s$ after expiration of a $50\,s$ warm-up period. Average the power samples. Round the average to the closest digit on the AC power meter.

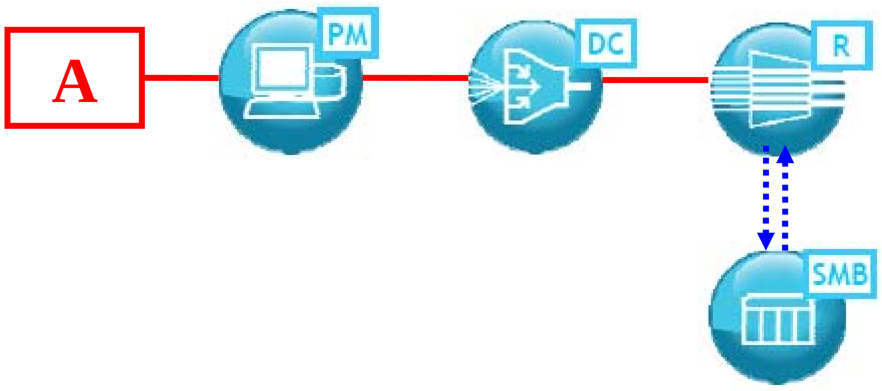

Figure 38. Testbed setup for Experiment R.6 (1000BASE-TX SFP, 10Mbps and 1000Mbps).

Results

Deleted, proprietary.

7. *Possible Refinements of the Model for Energy Profiling*

In this section we present results that offer indications for possible enhancements of our reference model for energy profiling of network equipment. Far from dismissing the importance of these results, we do not make them integral part of the reference model because of relevant limitations in the experimental process that produced them.

7.1 Effects of Packet Size

In network systems with extended packet processing capabilities, in particular those that handle packets based on the contents of packet headers at layer 3 and above, one may expect that the frequency of packet processing events has direct impact on the power consumption of the system.

For verification of this conjecture, we first run experiments with SUT2, sending unicast traffic at saturation rate from one port to another, and then varying the size of the Ethernet frames from a minimum of 64 bytes to a maximum of 1518 bytes. Since only the SMB traffic generator offers easy ways to automate the execution of experiments with gradually increasing packet sizes and traffic rates, we use the 10/100BASE-TX ports of the SMB to run experiments with the integrated 10/100BASE-TX ports of SUT2, and then the 1000BASE-SX 1GbE ports of the SMB to run experiments with the 1GbE ports of SUT2 loaded with 1000BASE-SX SFP's. In each experiment we use one port for receiving traffic from a generator port on the SMB and one port for forwarding traffic to a sink port on the SMB.

In Figure 39 we plot the results obtained with 10/100BASE-TX ports configured at $10\,Mbps$. In Figure 40 we plot results obtained with 10/100BASE-TX ports configured at $100\,Mbps$. Finally, in Figure 41 we plot results obtained with 1000BASE-SX ports configured at $1\,Gbps$.

Figure 39. Deleted, proprietary.

Figure 40. Deleted, proprietary.

Figure 41. Deleted, proprietary.

The plots show that the packet size does affect the power consumption of a port, with a fairly linear relationship between packet arrival rate and power consumption. Accordingly, the linear model of Equation (4), where the traffic-dependent parameters are based on bit-rate metrics, should be expanded as follows to incorporate packet-rate metrics:

$$P_S = P_C + \sum_{i=1}^{N_L} P_{L,i} + \sum_{j=1}^{N_P}\left[P_{0,j} + P_{E,j} + T_j \cdot (\rho_{IN,j} + \rho_{OUT,j}) + S_j \cdot \left(\sigma_{IN,j} + \sigma_{OUT,j}\right)\right], \qquad (5)$$

where S_j is the *packet saturation power* of port j at its configured rate of operation, measured with minimum-size packets, and $\sigma_{IN,j}$ and $\sigma_{OUT,j}$ are the input and output *packet saturation loads* of port j, expressed as the ratio between the actual rate in packets per second and the maximum packet-per-second rate supported by the port. Compared to Equation (4), the definition of T_j is more narrowly defined as the saturation power of port j with maximum-size packets.

Properietary material deleted.

Figure 42. Propietary, deleted.

Figure 43. Propietary, deleted.

Execution of the experiments with SUT3 yields similar results as those obtained with SUT2 (see Figure 44). Instead, with SUT1 the differences between the power measurements with small and large packets are negligible. This behavior is aligned with our premise to the experiments with different packet sizes, because SUT1 does not include hardware for the inspection and processing of packet headers at layer 3 and above.

Figure 44. Propietary, deleted.

Overall, the model of Equation (5) appears plausible and more comprehensive than the model of Equation (4). However, in Section 6 we maintain the tables obtained for the model of Equation (4) because of the limited set of port types for which we can estimate the parameters of Equation (5).

7.2 Non-Linear Behaviors

A small portion of experiments with SUT1, SUT2, and SUT3 showed non-linear behaviors whose inclusion in the models of Equation (4) and (5) is not trivial. Given the quantitative marginality of the effects of those behaviors, we decided not to include their characterization in our energy profiling models to preserve the linearity of those models.

7.2.1 SUT1 Cooling Fan

In the execution of the power measurement experiments with SUT1, we observe that in configurations with a large number of ports handling traffic, after some time the power consumption suddenly increases. We explain this increase with the activation of a cooling fan that only occurs when the system temperature exceeds a given threshold.

SUT2 and SUT3 do not exhibit the same behavior because their cooling fans appear to be running at all times.

7.2.2 SUT2 and SUT3 Throughput Cap

Both SUT2 and SUT3 appear to have an aggregate throughput limitation around $200\,Mbps$. As a consequence, as the aggregate input load grows above $200\ Mbps$ the growth of the power contribution per port slows down. This is particularly evident in Figure 41, where the rate of increase of the power consumption is lower (about half) than the same rate between $100\ Mbps$ and $200\ Mbps$. We explain the halving of the power growth rate with the fact that, in the specific system configurations that we selected for our experiments, for aggregate rates above $200\ Mbps$ only the input load keeps growing, while the output load is kept stable by the increasing amount of packets lost internally. The observation that the slope of the power consumption growth is halved when the systems are in the packet-drop region corroborates our simplifying assumption that input and output saturation powers are identical for a given port.

8. Conclusions

In summarizing the results of the EEDNRA task called Energy Profiling of Network Elements, we partition our conclusions in two distinct sets. The first set collects our findings with respect to energy profiling models and measurement methodologies. The second set focuses on the characterization of the tested systems with respect to the applicability of rate adaptation techniques at different timescales.

8.1 Energy Profiling Methodology

Our most important result with respect to energy profiling is the formulation of a comprehensive linear model for network elements with a broad range of different characteristics. While we have followed the model of Equation (4) for reporting our results in the tables and plots of Section 6, we have collected preliminary evidence that the model of Equation (5) can be even more accurate. This latter model should be further validated with future measurement experiments. Both models represent substantial enhancements over previously proposed models, because to the best of our knowledge they are first in the literature to explicitly capture the power cost of loading and enabling individual ports. Equation (5) effectively extends the model of Equation (4) to include the power contribution of the packet size distribution. This enhancement is particularly important in network elements with massive packet processing capabilities at layer 3 and above.

From a methodology perspective, our experience shows how critical it is to obtain direct measures of the target parameters as opposed to estimating them through indirect measurements. This is particularly true with respect to measuring the power consumption of a system with DC power supply (a DC power meter applied to the DC power supply wires would be more accurate than our AC power meter applied to the AC power supply of the DC module), and with respect to measuring the power contribution of individual ports (ideally, all ports should be attached to respective traffic generators/sinks and fully loaded in both directions for traffic-based measurements: the more peering devices are available for the ports of the system, the more accurate becomes the estimation of the parameters of the linear model). Because of our indirect collection of DC power measurements with two of the systems and because of the limited number of loaded ports in most of our experiments, we expect the inaccuracy of our estimation of the port parameters in the reference linear model to be generally high, and highest in the case of SUT2 and SUT3. The repetition of the experiments under ideal measurement conditions (with one traffic generator/sink per port and an accurate DC power meter) would provide a brute-force quantification of the error affecting our current estimates.

Another direction for improvement in energy profiling is the inclusion of nonlinear components such as thermostat-controlled cooling fans and aggregate-throughput caps in the reference model.

8.2 Rate Adaptation in Network Elements

Despite the relative inaccuracy of some of our measurements, especially those that apply to port parameters, our energy profiles still provide clear indications about the compatibility of current equipment with network-wide energy optimizations with FTRA techniques and the need for the introduction of PTRA capabilities in network hardware.

Our results show that modular systems like SUT3 currently offer larger opportunities for energy savings than monolithic systems like SUT1 and SUT2. However, these savings can only be achieved by physically removing unused line cards and port SFP's. In such equipment, the introduction of low-power sleep-modes at the card and port level would be extremely beneficial for utilization in conjunction with FTRA techniques. Similarly, the introduction of PTRA capabilities in the data-path hardware components of each module would help establish proportionality between traffic load and power consumption at times when the modules are actively in operation.

In monolithic systems, gaining remote control over the power states of the ports would be barely beneficial. The problem in this case is that most of the power is consumed in hardware components that are in full operational mode even when only few ports are active. The introduction of modular designs for such components and the application to each of them of short-timescale PTRA techniques, so that power consumption becomes a function of the aggregate traffic load, could radically improve the current picture.

One final observation emphasizes the importance of applying appropriate power supplies to the equipment in use. In our experiments, our DC module, designed to supply power to multiple systems, was operating in a region that was far from the one with the highest efficiency. This made the power dissipation of the DC module, not included in our linear models, extremely high.

9. References

[Antonak] S. Antonakopoulos, S. Fortune, L. Zhang, "Power-aware Routing with Rate-adaptive Network Elements," 3rd International Workshop on Green Communications (IEEE GreenComm3), Miami (FL), December 2010.

[Barroso] L.A. Barroso and U. Holze, "The Case for Energy-Proportional Computing," IEEE Computer, 40:12 (2007), 33–37.

[Benini] L. Benini, P. Siegel, G. De Micheli, "Saving power by synthesizing gated clocks for sequential circuits," IEEE Design & Test of Computers, Volume 11, Issue 4, 1994.

[Chabarek] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, S. Wright, "Power Awareness in Network Design and Routing," Proceedings of IEEE Infocom 2008, pp. 1130-1138, April 2008.

[ECR] ECR Initiative, "Network and Telecom Equipment—Energy and Performance Assessment," Draft 3.0.1, December 14, 2010.

[Francini] A. Francini, D. Stiliadis, "Performance bounds of rate-adaptation schemes for energy-efficient routers," Proc. of IEEE HPSR 2010.

[Mahadevan] P. Mahadevan, P. Sharma, S. Banerjee, P. Ranganathan, "A Power Benchmarking Framework for Network Devices," Proceedings of the 8th International IFIP-TC 6 Networking Conference, May 11-15, 2009, Aachen, Germany.

[MEF] Metro Ethernet Forum, "Metro Ethernet Services Definitions Phase 2," Technical Specification MEF 6.1, April 2008.

[Nedevschi] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing Network Energy Consumption via Sleeping and Rate Adaptation," Proc. of 5th USENIX Symposium on Networked Systems Design and Implementation (San Francisco, CA, 2008), pp. 323–336.

[Octave] GNU Octave. <http://www.delorie.com/gnu/docs/octave/octave_toc.html>

[Rossi] D. Rossi, A.P. Bianzino, J.-L. Rougier, C. Chaudet, F. Larroca, "Energy-Aware Routing: a Reality Check," 3rd International Workshop on Green Communications (IEEE GreenComm3), Miami (FL), December 2010.

[Tamm] O. Tamm, C. Hermsmeyer, A.M. Rush, “Eco-Sustainable System and Network Architectures for Future Transport Networks,” Bell Labs Technical Journal 14(4), 311–328 (2010), February 2010.

.

Selection of a Rate Adaptation Scheme for Network Hardware

Andrea Francini
Alcatel-Lucent Bell Laboratories
Mooresville, NC (USA)
Email: andrea.francini@alcatel-lucent.com

Abstract—**Rate adaptation is a family of technologies driven by the expectation that large energy savings can be achieved in packet networks by dynamically adjusting the capacity of network components to the load that they are required to sustain. Depending on the scope of their application, whether at the network, system, or circuit-pack level, specific instances of the rate adaptation concept differ widely by their timescale of operation. In this paper we focus on packet-timescale rate adaptation (PTRA) techniques, which apply to individual traffic processing chips in the circuit packs of network systems. We look at the field of available options for PTRA implementation, whose behavior has been so far well characterized only in single-device applications, and compare their performance in realistic multi-device configurations. We find that the sleep-state-exploitation (SSE) scheme, which only adds a sleep state to the ordinary full-capacity state, offers the best compromise between energy savings and the packet delay degradation that PTRA unavoidably introduces. We then study the effects of SSE on the utilization of a bottleneck link in the data path of a set of TCP connections. We recognize the need for buffer size adjustments to compensate for the presence of SSE devices along the data path, in measures that largely depend on the buffer management policy that regulates access to the bottleneck queue. The adjustment needed with the recently defined periodic early detection (PED) scheme is considerably smaller than that required by the conventional tail-drop policy and almost negligible in configurations of practical relevance. All results that we present consistently support the conclusion that the key technological enabler for the widespread adoption of PTRA will be the availability of SSE devices with a very low-power sleep state and state transition time well within the microsecond range.**

Keywords-energy efficiency; rate adaptation; sleep mode; buffer management

I. INTRODUCTION

In packet networks, the term rate adaptation designates a broad set of technologies that aim at establishing a linear relationship between sustained workload and power consumption. The ideal power-rate function has minimum slope and near-null power when no traffic is present [1]. To approximate such behavior, rate adaptation schemes typically provide the systems that they control with a discrete set of operating states, where each state maps a fixed traffic processing rate onto a respective power consumption level.

The scope of the control exercised by a rate adaptation scheme can range from large subsets of network links and nodes [2], [3] to individual sections of a single traffic processing chip [4]. Hence, for the sake of clarity we partition rate adaptation techniques based on their timescale of operation, which is defined by the time needed to complete a transition between states and ultimately depends on the size of the targeted system. *Flow-timescale rate adaptation* (FTRA) techniques control the state of network links and nodes based on expected or measured trends in traffic demands between network endpoints [2], [3]. FTRA state transitions involve network signaling and system-level power cycles, so their timescale ranges from seconds to minutes. *Packet-timescale rate adaptation* (PTRA) techniques adjust the clock frequency and supply voltage of data-path hardware components to locally maintained workload indicators such as queue lengths and traffic arrival rates [5], [6]. The timescale of PTRA state transitions ranges from microseconds to milliseconds depending on the underlying integrated circuit technology. Finally, *bit-timescale rate adaptation* (BTRA) also applies to data-path hardware components. Compared to PTRA, BTRA transitions are much faster to execute (nanoseconds) because they only involve control of the system clock (e.g., by gating of the clock signal), at the expense of reduced power savings.

The high speed of the state transitions trivializes the BTRA control algorithm, which gates the clock signal as soon as there is no packet to be processed and restores it immediately after a new packet arrives. With PTRA, instead, the execution of a state transition may span over multiple packet times and involves an added energy cost. As we have shown in [6], such state transition overheads require the PTRA control algorithm to keep the same operating state for at least a *minimum hold time* after its selection, so that the total time spent in the execution of state transitions falls below a fixed fraction of the total operating time. Unfortunately, the minimum hold time also introduces extra delay along the network data path.

PTRA easily complements rate adaptation techniques at the other timescales. FTRA schemes that route traffic based on network-wide energy-optimization criteria rely on the power-rate relationship established by PTRA for construction of their cost functions [2], [3]. BTRA and PTRA techniques can coexist by application to different portions of the same system. The overall energy-saving yields of PTRA grow with the peak-to-average ratio of traffic demands, and therefore with the distance of its domain of application from the center of the network core. Still, any network node, even in the core, can derive important energy benefits from the use of PTRA, whether in isolation or combined with other techniques.

This work was completed with the support of the U.S. Department of Energy (DOE), award no. DE-EE0002887. However, any opinions, findings, conclusions and recommendations expressed herein are those of the authors and do not necessarily reflect the views of the DOE.

Many PTRA schemes have been proposed in the past [5], [6], but it remains unclear which approach best resolves the trade-offs that exist between energy savings, total extra delay, and incremental deployability [6]. Together with the lack of confidence in the transparency of TCP performance to the insertion of PTRA in the data path, such uncertainty contributes to the persistence of heavy skepticism on the prospects of widespread inclusion of PTRA in future network systems.

The contributions of this paper subdue both reasons for skepticism. First, by comparison of energy-saving and delay performance in practical deployment scenarios, we conclude that *sleep-state exploitation* (SSE) presents substantial advantages over its most qualified contenders, namely two schemes from the *hybrid rate adaptation* (HRA) class [6]. A small modification of the original SSE algorithm enables large reductions of extra delay accumulation in traffic processing lines that include multiple rate-adaptive devices (as in the line cards of modular network elements) without requiring signaling between the devices. The elimination of signaling, which enables multi-vendor board designs without a dedicated standard, does not appear to be obviously attainable with HRA schemes. Second, we study the impact of the introduction of SSE in the data path of TCP flows when two different buffer management policies are applied to the network bottleneck queue, concluding that the presence of SSE devices is virtually unperceivable when the total extra delay that they add is compatible with the delay budget of real-time applications.

The paper is organized as follows. In Section II we summarize previously published results on PTRA. In Section III we look at the effects of typical deployment conditions on the energy-saving and delay performance of selected PTRA schemes. In Section IV we study the impact of SSE on the throughput of TCP flows at a bottleneck queue that implements the conventional tail drop policy and an active queue management scheme of recent specification called *periodic early detection* (PED) [7]. We present final conclusions in Section V.

II. PTRA Definitions and Known Results

A PTRA device receives and forwards data packets over one or more interfaces and is capable of adjusting its operating state to the traffic load it is offered. Network elements like switches and routers can include one or more PTRA devices in their data-path cards. The PTRA device is arranged with a finite set $S = \{s_k, k = 1 \ldots K\}$ of discrete operating states, where each state s_k corresponds to fixed values of clock frequency and power-supply voltage. Every frequency value yields a unique traffic processing rate. Every state has two power consumption levels, for times when the device processes traffic (*active power*) or stays idle (*inactive power*). We assume that the time δ needed to complete a state transition is fixed and independent of the two states involved, and that during a state transition the device consumes the average of the inactive powers for the starting and ending states.

The notion of PTRA domain [6] enables the incremental deployment of PTRA in network nodes, avoiding the need of standardized protocols for global coordination. As shown in Figure 1, a PTRA domain includes an input interface I, a processing unit P, and an output interface O. The domain uses the state of input queue Q_I to set the rate at which interface I obtains packets from Q_I, unit P processes the packets, and interface O delivers them to output queue Q_O, independently of the rate at which the RA domain immediately upstream (downstream) writes (reads) packets into Q_I (out of Q_O). In practice the processing unit can consist of one device or multiple devices (in the latter case, a single controller sets the operating state for all devices). PTRA domains use pre-existing buffer stages in the data path to isolate the control of their operating states and thus eliminate the need for inter-domain coordination. From a hardware design perspective, the delineation of a PTRA domain does not introduce new memories or buffering stages, since a buffering stage is needed anyway for clock isolation between adjacent clock domains. All the definitions and results overviewed in this section apply to a PTRA domain observed in isolation and independently of its internal components.

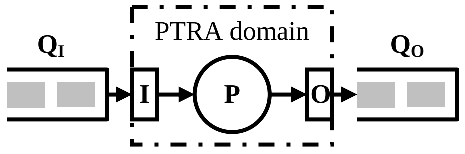

Figure 1. The PTRA domain abstraction.

The design of a PTRA scheme develops along the following dimensions, further discussed immediately below [6]: (1) subdivision of the time axis into operating intervals; (2) choice of the reference power-rate curve; and (3) specification of the state-setting policy.

Operating Intervals—Since state transitions increase energy consumption and packet delays, the total time spent in their execution should be curbed. This is achieved by imposition of a minimum hold time $h > \delta$ between consecutive state transitions. The minimum hold time is the dominant parameter in the energy-saving and added-delay metrics of all PTRA schemes [6].

Power-Rate Curves—A power-rate curve maps traffic processing rates onto power consumption levels. The *active power-rate* (A-PR) curve of a PTRA domain defines the relationship between traffic processing rate and active power and drives the design of the PTRA scheme. The *inactive power-rate* (I-PR) curve maps processing rates onto inactive power levels. Figure 2 illustrates the A-PR curves of the *sleep-state exploitation* (SSE) scheme, which only provides a full capacity state $s_C = (C, P_C)$ and a low-power sleep state $s_0 = (0, P_0)$, and of the *hybrid rate adaptation* (HRA) scheme, which combines the sleep and full-capacity states with an extended set of intermediate *rate-scaling* states $\{s_1, \ldots, s_V\}$ enabled by *dynamic voltage and frequency scaling* (DVFS) designs [8], [9], [10]. Compared to a plain rate-scaling scheme without sleep state, the HRA curve enables linear scaling of the energy consumption in the lower portion of the load range, where DVFS cannot reduce further the supply voltage (see [6] for details on deriving the optimum A-PR curve for HRA).

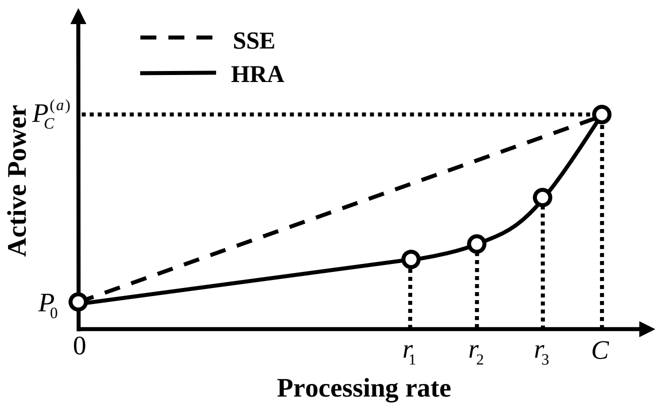

Figure 2. Active power-rate curves of SSE and HRA schemes.

State-Setting Policies—The state-setting policy is the algorithmic core of any PTRA scheme. It sets the time and destination state of each state transition. In the remainder of this section we recall the definitions of classes of state-setting policies that are capable of enforcing deterministic bounds on the added extra delay and are amenable to the computation of upper and lower bounds on the average power consumption. We provide formulas for the delay bound and show a sample plot for the energy bounds.

Definition 1 (Backlog Clearing Policy)—At time t_0, when (i) the PTRA domain is not processing a packet, (ii) the minimum hold time has expired since entering the current state, (iii) the current processing rate is less than maximum $(r(t_0^-) < C)$, and (iv) the current processing rate is not sufficient to clear the current queue backlog by the next expiration of the minimum hold time $(r(t_0^-) < Q(t_0)/h)$, a *backlog clearing* (BC) policy picks the new processing rate from the set of available rates that can clear the queue backlog by the closest possible end of the next operating interval $(t_1 = t_0 + h)$: $r(t_0) \in \{r_k : s_k \in S \wedge r_k \geq Q(t_0)/(h-\delta)\}$. If no available rate can clear the backlog, the policy chooses the full capacity of the PTRA domain: $r(t_0) = C$.

A BC policy can be combined indifferently with SSE and HRA power-rate curves. The following bound holds on the extra delay added by an HRA domain with BC policy [6]:

$$d_{HRA} < \max(d_{Max}, d_{Max}^{(h)}), \tag{1}$$

where $d_{Max}^{(h)} = h + \delta + L_{Max}/r_{Min}$ is the *maximum delay under heavy load* and $d_{Max}^{(l)} = h[2 - r_{Min}/C] + \delta \cdot r_{Min}/C + L_{Max}/C$ is the *maximum delay under light load* (r_{Min} is the minimum non-null rate and L_{Max} is the maximum packet size). The delay bound is generally lower for an SSE domain with identical parameters:

$$d_{SSE} < h + \delta + L_{Max}/C. \tag{2}$$

The two types of *receding backlog clearing* (RBC) policies that we define next specify distinct behaviors for times when a PTRA domain completes processing of the last packet in the queue.

TABLE I.
OPERATING STATES FOR THE SAMPLE PLOT WITH ENERGY BOUNDS

STATE	RATE [Gbps]	ACTIVE POWER [W]	INACTIVE POWER [W]
s_0	0 (sleep)	0.029	0.029
s_1	5	0.282	0.226
s_2	6	0.341	0.273
s_3	7	0.435	0.348
s_4	8	0.571	0.457
s_5	9	0.756	0.605
s_C	10	1.000	0.800

Definition 2 (Fast-Receding Backlog-Clearing Policy)—A *fast-receding backlog clearing* (FRBC) policy is a BC policy that always triggers a transition to the sleep state if it finds the input queue empty at the end of a packet transmission, irrespective of the time already spent in the current state.

Definition 3 (Slow-Receding Backlog-Clearing Policy)—A *slow-receding backlog clearing* (SRBC) policy is a BC policy that always triggers a transition to the state immediately below the current state if it finds the input queue empty at the end of a packet transmission or at the end of a state transition, irrespective of the time already spent in the current state.

The distinction between FRBC and SRBC policies is only relevant to HRA domains: from now on, the two acronyms refer with no exception to HRA domains. Every reference to SSE that follows implies a generic RBC policy.

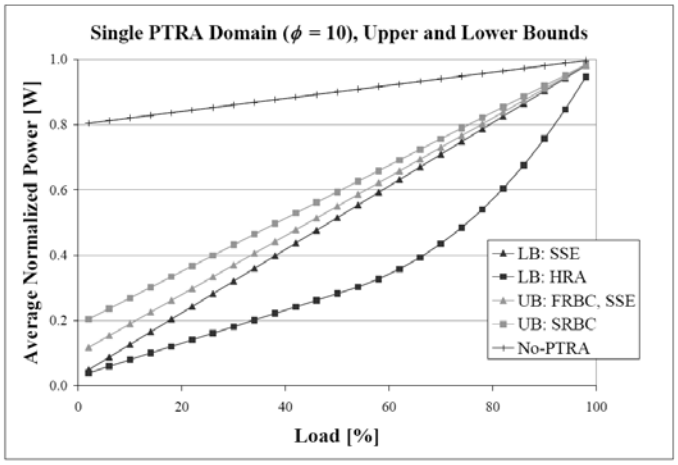

Figure 3. Lower and upper bounds on normalized power consumption in a single PTRA domain (SSE, S-RBC, F-RBC).

The RBC policy definitions enable the calculation of closed-form expressions for upper and lower bounds on energy consumption as a function of the ratio $\varphi = h/\delta$ between the minimum hold time and the transition time. It is shown in [6] that the absolute values of δ and h, which dominate in the delay-bound formulas (1) and (2), have no impact on the energy bounds, and that $\varphi = 10$ yields the best compromise between energy and delay bounds. Figure 3 plots the energy bounds for SSE and HRA domains whose power states are listed in Table I (the table entries are obtained by interpolation and normalization of the operating states of the Intel Pentium® M Processor [11]). In practical deployment scenarios, the lower bounds of Figure 3 are approached by steady traffic patterns, whereas plots near the upper bounds result from traffic patterns with a high degree of scattering for rate samples collected over minimum-hold-time intervals.

By plain observation of the energy bounds of Figure 3, FRBC should be the PTRA scheme of choice, because it combines the best lower and upper bounds. The evidence shown in the next section supports a quite different conclusion.

III. PTRA Performance in Practical Scenarios

We use the ns-2 simulation platform [12] to compare the energy-saving and delay performance of SSE, SRBC, and FRBC in three scenarios of immediate practical relevance.

A. Single PTRA Domain with CBR and Pareto Traffic

In the first scenario we look at the effects of different traffic patterns on the energy-saving and delay performance of a single PTRA domain. We feed the domain with a stream of fixed-size packets ($1000\,bytes$) that in different instances of the experiment are generated by a *constant bit rate* (CBR) source and by a Pareto source with $0.1\,ms$, $1\,ms$, and $10\,ms$ average on and off times. We expect the CBR pattern to test the tightness of the lower bounds on energy consumption plotted in Figure 3, especially for the SRBC and FRBC schemes, and for the Pareto patterns to challenge the upper bounds. We use different values of the average on/off times for the Pareto source to assess the sensitivity of the energy-saving performance to the ratio between the timescales of the traffic bursts and of the PTRA operating intervals (in all the experiments we configure the PTRA domain with $\delta = 0.1\,ms$ and $h = 1\,ms$). We compute the average normalized power using the entries of Table I.

Figure 4 compares the average power that SSE, SRBC, and FRBC consume under CBR traffic with the respective lower bounds. The experimental results with SRBC match closely the corresponding lower bound. The same is not true quite evidently for FRBC and to a lesser extent for SSE. We explain the FRBC discrepancy with the immediate transition to the sleep state that the state setting policy mandates every time it finds the input queue empty: the FRBC domain pays extra energy for the wide oscillations between the sleep state and the state with processing rate immediately above the CBR arrival rate. With the SRBC policy, instead, the arrival of new packets before completion of the first downward state transition allows the domain to oscillate between contiguous states. Given the convex nature of the PR curves, narrower state oscillations induce lower energy consumption. In the SSE case, the distance between the experimental curve and the lower bound is caused by the duration of the state transitions (the curves get closer as the value of δ becomes smaller).

Figure 5 compares the average power of SSE, SRBC, and FRBC under Pareto traffic with the respective upper bounds. The average on/off time at the source is $10\,ms$, ten times larger than the minimum hold time h. The three experimental curves overlap completely in the upper portion of the load range, whereas differences can be appreciated between SSE and the HRA schemes at loads below 50%. In the plots obtained from experiments with average on/off period equal to $0.1\,ms$ and $1\,ms$ (shown in the Appendix) we observe that the load level at which the SSE curve joins the other two curves increases as the average on/off period decreases. Figure 5 also shows that the SRBC experimental curve consistently remains below the joint upper bound of SSE and FRBC, even though its upper bound is higher. This is because the SRBC upper bound is the product of a very unlikely periodic on-off traffic pattern that repeats itself with duty cycle specifically tailored to the values of δ and h. Under more realistic traffic patterns, even very adverse ones, SRBC can safely be expected to match or outperform FRBC.

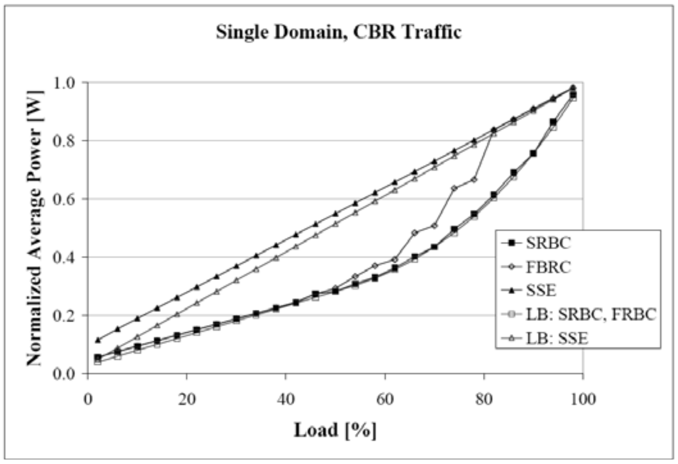

Figure 4. Average power consumption with CBR traffic, compared to lower bounds.

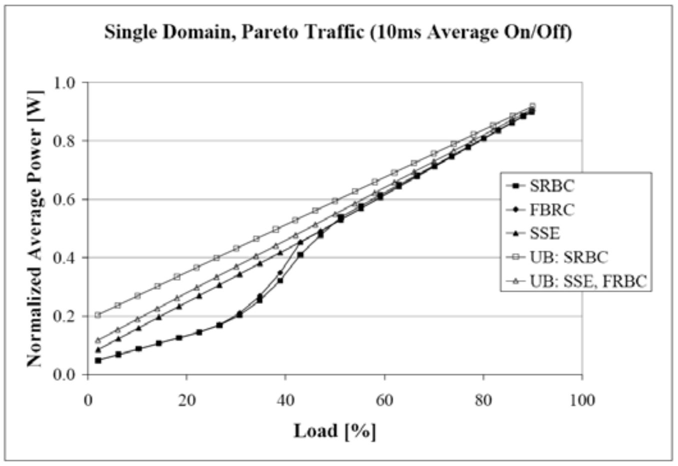

Figure 5. Average power consumption with Pareto traffic, compared to upper bounds (average on/off time larger than the minimum hold time).

B. Line of Two PTRA Domains with SSE First

Our next set of experiments focuses on a simple topology of PTRA domains that will likely become commonplace after PTRA devices start appearing in network equipment. The topology, shown in Figure 6, consists of a *line* of two PTRA domains where the type of the first domain is fixed (SSE) and the type of the second domain can be SSE, SRBC, or FRBC. (We define a line as a sequence of PTRA domains that handle the same set of packet flows, with no multiplexing or demultiplexing of flows in between the domains.)

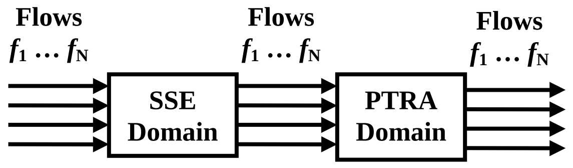

Figure 6. Line of two PTRA domains.

The line topology of Figure 6 describes well the line card of a router or layer-2 switch whose network or backplane adapters comply with the IEEE 802.3az standard for energy-efficient ethernet [13]. The SSE domain models the link behavior specified by the standard, while the generic PTRA domain represents the next processing stage in the line card data path. (The IEEE 802.3az task force also considered a multi-state scheme more similar to HRA in early discussions, but eventually decided to drop it.)

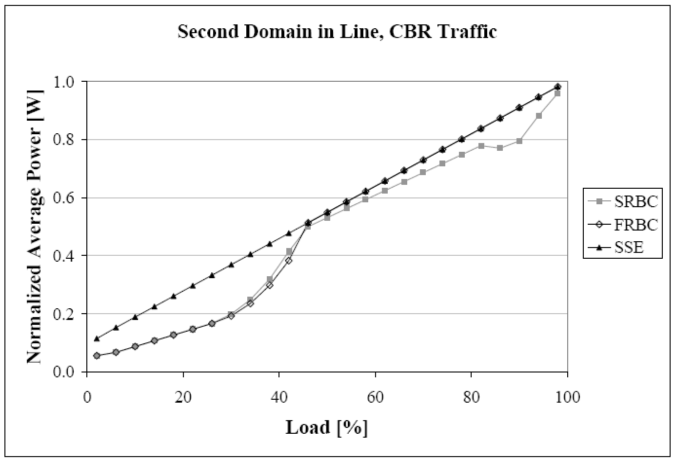

Figure 7. Average power consumption in the second domain of a line with SSE domain in first position (CBR input traffic).

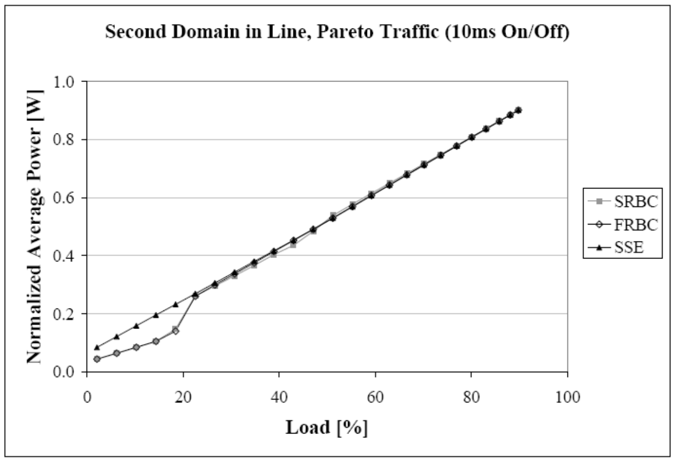

Figure 8. Average power consumption in the second domain of a line with SSE domain in first position (Pareto traffic with 10ms average on/off time).

Since an SSE domain only operates at full capacity when it is not sleeping, the profile of its output traffic is always on/off, irrespective of the profile of the input traffic. Accordingly, we expect the energy-saving performance of the second domain to show minimal sensitivity to the characteristics of the traffic at the ingress of the line, with experimental power plots closer to the upper bounds than to the lower bounds.

The plots of Figures 7 and 8 present results for the two extremes of the input traffic spectrum. With CBR traffic at the ingress of the two-domain line (Figure 7), FRBC and SSE fully overlap in the upper half of the load range, whereas SRBC maintains a slight advantage. In the lower half, the separation between SSE and the two HRA schemes is more evident. With Pareto traffic at the ingress of the line (Figure 8), the three schemes have identical performance at all load levels above 20%. Only minor differences can be observed at lower loads.

On one hand, the results of Figures 7 and 8 confirm that in deployment scenarios with energy-efficient links of the kind specified in the IEEE 802.3az standard the difference in energy-saving performance between SSE and the HRA schemes becomes marginal. On the other hand, the results highlight the more general principle that the HRA schemes can perform at the best of their capabilities only in isolation or in lines of homogeneous domains, the latter being a condition that is hard to enforce in line card environments that typically assemble hardware components from multiple vendors. The HRA reliance on homogeneous multi-domain designs becomes even more critical in the scenario that we study in the next subsection.

C. Delay Accumulation in Multi-Domain Lines

The definition of PTRA domain, which is instrumental to the incremental deployability of PTRA capabilities in network equipment, implies that the total delay added to the data path by a line of PTRA domains is the sum of the delays added by the individual domains of the line. This linear accumulation of extra delay may compromise the ability to deploy PTRA pervasively in every portion of the network, simply because the total delay added by PTRA along the average end-to-end network path may exceed the delay budget of network applications with stringent delay requirements. For a given PTRA technology, which yields specific values of δ, h, and maximum delay (see (1) and (2)), the challenge is to achieve sub-linear accumulations of extra delay wherever possible.

The introduction of signaling for state coordination between contiguous PTRA domains is one of the available options for limiting the accumulation of extra delay. An upstream domain could share information about its operating state with the domains that immediately follow it in the data path. While inter-system coordination would require the standardization of a signaling protocol, proprietary signaling mechanisms would be sufficient for intra-system coordination. However, in all cases the adoption of explicit state coordination between PTRA devices adds complexity and conflicts with the incremental deployability property that makes PTRA domains appealing. We strongly prefer a solution that effectively cuts the delay accumulation without altering the mode of operation of each PTRA device.

Multi-domain PTRA lines are easy to delineate out of the sequences of traffic processing stages found in the line cards of all kinds of network systems (ranging all the way from wireless and wired access nodes up to core IP routers). The SSE scheme, with its minimum number of operating states, is very well suited for functional adjustments that reduce the accumulation of extra delay in such lines of PTRA domains. As opposed to multi-state HRA schemes, SSE can only choose one active state. Therefore, it does not need to wait for the accumulation of packets in the input queue to make the most appropriate state selection. If traffic is guaranteed to arrive to an SSE domain in bursts of back-to-back packets separated by silence intervals of duration never shorter than h, the SSE domain does not consume more energy if it no longer waits for an extra time h before processing the first packet of a new burst. Instead, the domain can instantly start the transition to the full-capacity state and process the packet at the end of the transition, because the packet is followed either by a new packet or by a new silence period of duration not shorter than h. We call *instant sleep-state exploitation* (ISSE) domain an SSE domain that immediately triggers the transition to the full-capacity state when its empty input queue receives a packet, without waiting for the expiration of a minimum hold time.

We consider now the example of a packet processing pipeline in a router line card, shown in Figure 9. Taking advantage of the small buffering stages that exist between them, we map the network adapter, the packet processor, the traffic manager, and the switch fabric adapter of the line card onto respective PTRA domains. Since the four processing stages handle the same traffic flows, we can qualify them as a line of PTRA domains.

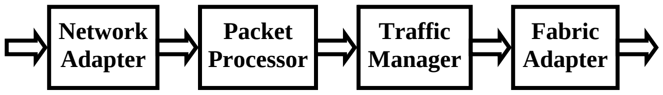

Figure 9. Traffic processing pipeline of a router line card.

We run simulation experiments with the usual traffic patterns to compare the delay and energy-saving performance of five different combinations of PTRA domains, labeled as follows: (1) FRBC: four FRBC domains; (2) SRBC: four SRBC domains; (3) SSE: four SSE domains; (4) MIX: one SSE domain followed by three ISSE domains; and (5) ISSE: four ISSE domains.

In Figures 10, 11, and 12 we plot the average power consumption, average delay, and maximum delay measured with the above combinations of PTRA domains when the input traffic has a Pareto profile with average on and off periods of $0.1\,ms$. Figure 10 shows that the replacement of the last three SSE domains with three ISSE domains has no negative impact on the energy-saving performance of the line (the SSE and MIX curves are identical). Figures 11 and 12 reveal instead the substantial benefits of the same replacement with respect to the accumulation of delay.

It is easy to prove that, in a line formed by one SSE domain and $N-1$ ISSE domains, the total maximum delay is $d_{Max}^{MIX}(N) = h + N\,\delta$, which is smaller by $(N-1)\,h$ than the maximum delay with N identical SSE domains $(d_{Max}^{SSE}(N) = N\,(h+\delta))$, and larger only by h than the maximum delay with N ISSE domains $(d_{Max}^{ISSE}(N) = N\,\delta)$.

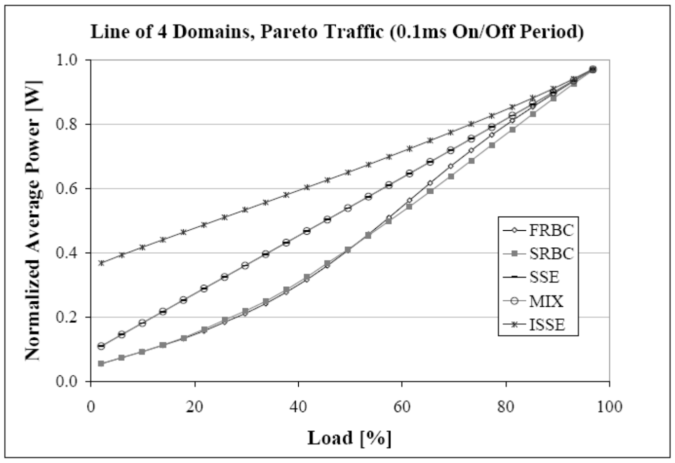

Figure 10. Average power consumption in a line of four PTRA domains.

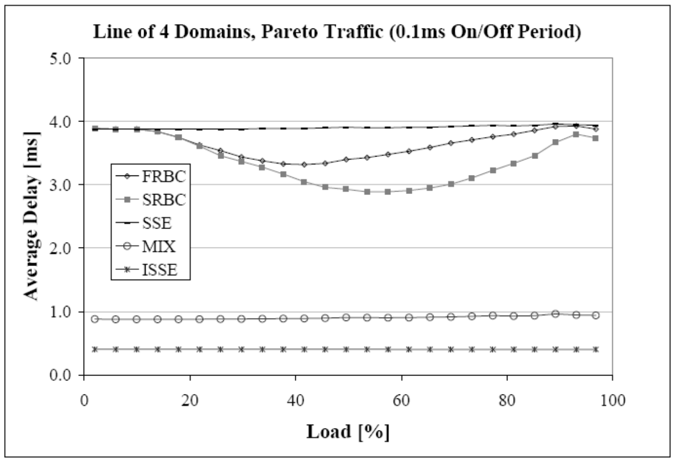

Figure 11. Average delay in a line of four PTRA domains.

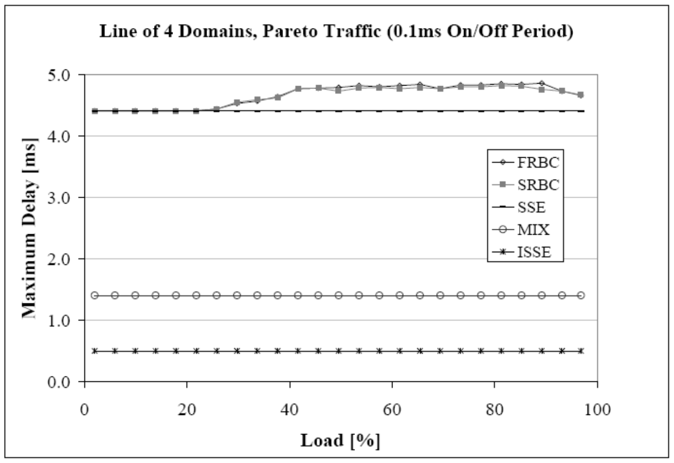

Figure 12. Maximum delay in a line of four PTRA domains.

The results of the experiments with Pareto sources with longer on/off periods, shown in the Appendix, corroborate the conclusion that the MIX line largely outperforms all other lines when both energy savings and added delays are considered. They also indicate that a line of four ISSE domains improves its energy-saving performance constantly as the average on/off period of the source increases. This outcome is not surprising, because longer on/off periods in the input traffic naturally extend the hold time for each of the two SSE states, making the artificial enforcement of a minimum hold time in the first domain unnecessary. Indeed, it is possible to eliminate the minimum hold time h altogether when the state transition time δ falls below a threshold that is defined by the burstiness characteristics of the incoming traffic. As a consequence, the removal of the minimum hold time is safe not only in BTRA domains, but also in packet-timescale SSE domains with state transition time not larger than the transmission time of the average packet.

IV. SSE Impact on TCP Traffic

The first proposals to apply rate adaptation techniques to network equipment were immediately met by concerns about the negative effects that rate adaptation may have on TCP traffic [14]. The argument is simple: since TCP is made to grab all the bandwidth that is available on a network link, how can it reconcile with a technology that tries to reduce the available bandwidth to the minimum necessary to sustain the current load? By the time a TCP source has recognized that it can increase its traffic generation rate to take possession of available extra bandwidth, rate adaptation may already have deprived the link of that extra bandwidth. In this section we approach the problem with exclusive focus on PTRA schemes. Having established in the previous section that SSE offers a better trade-off between delay and energy savings than the two HRA schemes, we look specifically at the effects produced by the presence of one or more SSE domains along the data path of a set of TCP connections.

We run ns2 simulations on the dumbbell network topology of Figure 13, where a set of TCP Reno connections share a portion of their data paths that includes a source aggregation node (SAN), $N \geq 1$ consecutive nodes that, in the forward path only, are capable of operation as SSE domains (SSD), a bottleneck node (BNN), and a sink distribution node (SDN). Assuming that the SSE domains can generally be located in distinct network nodes, with cross-traffic flows multiplexed and de-multiplexed at each node, we exclude the use of ISSE domains for minimization of the total extra delay. All shared links have $10\,Gbps$ capacity, with the exception of the bottleneck link between the BNN and the SDN, which runs at $1Gbps, 2Gbps, \ldots, 9Gbps$ in different experiments. The access and distribution links run at $1Gbps$. All links have negligible propagation delay, except for the access links between the TCP sources and the SAN, which define the *round-trip-time* (RTT) values for the respective flows. We refer to the queue in front of the BN link as the BNQ.

Out of the extensive set of traffic configurations that we have tried in our experiments, which varied by RTT distribution, by number of TCP flows in the primary path, and by number of cross-traffic flows at each network node, we present here results that we consider ideal for illustrating the mechanics of the interaction of PTRA with TCP traffic. Our representative scenario has 50 long-lived TCP flows, each with the same RTT of $20\,ms$, and no cross-traffic flows. We work with $N = 10$ SSD nodes. Unless otherwise specified, the state transition time of every SSE domain is $\delta = 1\,ms$ and the minimum hold time is $h = 10\,ms$. The SSE parameters are in the same order of magnitude as the RTT for best appreciation of the SSE impact on TCP dynamics; in practical applications we can expect the timing parameters of a single domain to be lower. For every TCP source the maximum size of the transmitter window is large enough to ensure that the flow is bottlenecked at the BN link and not at the source.

We collect results from the application of two distinct buffer management policies to the BNQ: the conventional *tail-drop* policy, where a packet is dropped when its arrival finds no space in the queue [15], and *periodic early detection* (PED), an *active queue management* (AQM) scheme of recent specification that enables dramatic buffer space reductions in router and switch designs [7]. We discuss the respective sets of experiments in the two subsections that follow.

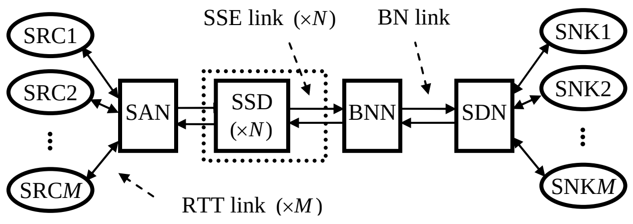

Figure 13. Network configuration for the simulation experiments.

A. *Experiments with Tail Drop*

To retain full link utilization, the tail-drop policy calls for sizing the BNQ buffer with the *bandwidth-delay-product* (BDP) rule: $B = C_{BN} \cdot \overline{\vartheta}$, where C_{BN} is the capacity of the bottleneck link and $\overline{\vartheta}$ is the average RTT over the set of TCP connections that traverse the bottleneck link [15]. We remark that $\overline{\vartheta}$ does not include the contribution of the BNQ to the RTT of the TCP connections.

The plot of Figure 14 shows steady-state link utilization levels measured at the BN link with different modes of operation of the SSE domains and different BNQ sizes. Consistently with the BDP rule, for every value of the bottleneck rate we have a different buffer size but the maximum queueing delay at the BNQ is fixed. As expected, when the SSE domains are forced to remain always in the active state (the *No SSE* curve) the link utilization stays close to 100%. When we restore the ordinary SSE operation, instead, the drop in link utilization is evident (see the *10 SSE, B = 20ms* curve). The throughput loss is caused by the increased RTT that the TCP connections experience when the SSE domains are in regular operation. We know from (2) that the delay added by the ten SSE domains is at most $10\,(\delta + h) = 110\,ms$. Conjecturing that the contribution of the SSE stages to the RTT is equal to their average added delay ($\overline{d} = 55\,ms$), we accordingly reconfigure the size of the BNQ: $B = 20 + 55 = 75\,ms$. As shown by the *10 SSE, B = 75ms* curve, the buffer size increase almost entirely recovers the original link utilization.

The experiments with tail-drop policy yield the following conclusions: (a) The SSE domains along the data path increase the end-to-end delay and therefore the RTT of the TCP connections. (b) The RTT increase is proportional to the total maximum added delay of all the SSE domains in the data path and therefore grows with the number of SSE domains in the data path; the ratio between the RTT increase and the total maximum added delay ranges between 0.5 and 1. (c) For mitigation of the RTT increase, so that the utilization of the bottleneck link is not reduced, the RTT increase must be included in the BDP formula for sizing the buffer of the BNQ. (d) The placement of the BNQ relative to the set of SSE domains is irrelevant to the quantification of the buffer size increase needed for avoiding losses of link utilization: the increase is the same if the BNQ is in the middle of the SSE domains on the forward path, and also if some or all of the SSE domains are in the return path.

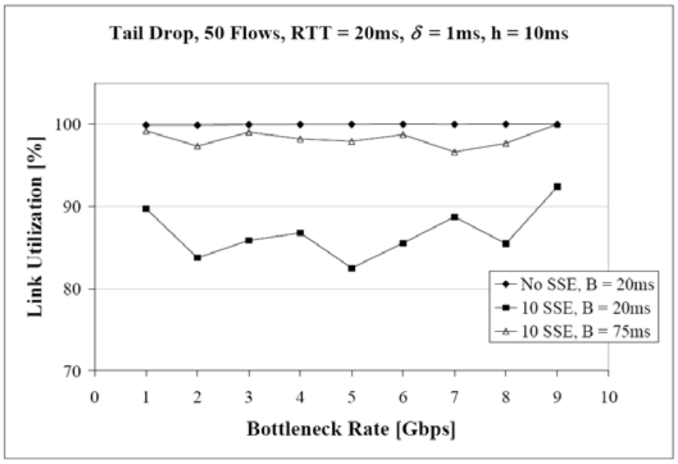

Figure 14. Utilization of the bottleneck link with tail-drop policy at the BNQ.

In practical deployments, the SSE domains handle all kinds of network traffic, not only TCP. This includes traffic with real-time requirements. In order for the presence of SSE domains to be accepted in the data path, their cumulative maximum added delay in each direction must be compatible with the end-to-end delay requirements of real-time network applications that share the same path. To this effect, a fixed delay budget, for instance $D_{SSE} = 50\,ms$, could be allocated each way for SSE deployment. With the SSE parameters of the simulation example, $50\,ms$ translate into ten SSE domains at most. With more realistic state transition times (in the microsecond range), the number of allowed SSE domains becomes much larger. Compliance of the deployed SSE domains with D_{SSE} translates into a bounded increase of the RTT for TCP traffic ($\Delta\overline{\vartheta} = 2\,D_{SSE} = 100\,ms$ in the example). Such a bounded increase, derived directly from the SSE delay budget for real-time applications, should be factored in the sizing of every network buffer that uses the tail drop policy to discard TCP packets when congested.

B. *Experiments with Periodic Early Detection*

Active queue management (AQM) schemes aim at reducing buffer occupancy and queueing delay by randomly dropping TCP packets long before the buffer fills up. When the few TCP connections involved in the early packet losses respond to them by slowing down their traffic generation rate, the queue length stabilizes around an average value that can be much smaller than the average queue length (AQL) of a tail drop queue. *Periodic early discard* (PED) is an AQM scheme of recent definition [7] that eliminates critical flaws of the better-known *random early detection* (RED) algorithm, present in both original [16] and revised formulations [17], [18].

PED replaces the drop probability curve of RED, which maps the instant value of the AQL onto a packet drop probability, with a packet drop rate that is maintained over extended periods of time while the AQL keeps moving. The packet drop rate is reassessed at fixed time intervals (e.g., every $500\,ms$), or earlier if the AQL shows signs of instability (see [7] for a detailed specification of PED). The new scheme is trivial to configure and in typical benchmarking scenarios yields substantial buffer size reductions (over 95%) compared to the BDP rule (e.g., $8\,MB$ vs. $300\,MB$ in front of a $10\,Gbps$ link). Here we look at the impact of SSE on the configuration of the PED scheme, knowing that PED shows very low sensitivity to the RTT of the TCP connections [7].

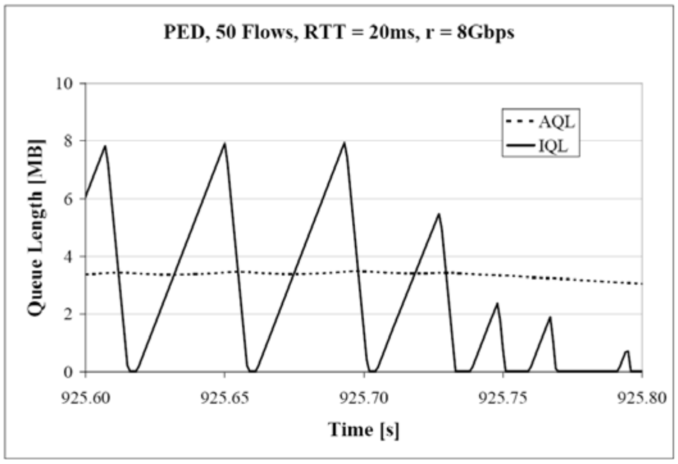

Figure 15. Short-term BNQ evolution with default configuration of PED parameters (200ms interval).

We repeat the experiments of the previous subsection after replacing the tail-drop BNQ with a PED BNQ of size $B = 8\,MB$ ($6.4\,ms$ maximum queueing delay at $10\,Gbps$) and default parameter configuration from [7]. We observe 100% link utilization when the sleep state is disabled in the SSE domains. When we re-enable the sleep state in just one SSE domain, the link utilization drops steeply as the bottleneck rate increases, hitting a 50% minimum at $8\,Gbps$. Figure 15, which plots the AQL and the instantaneous queue length (IQL) over a $200\,ms$ interval when the bottleneck rate is $r = 8\,Gbps$, explains the drop in link utilization: the presence of the SSE domains not only increases the RTT of the TCP connections, but also causes short-term queue length oscillations at the BNQ. The oscillations overflow the small buffer before a single early discard event takes place.

In Figure 15 the limited buffer size clips the oscillations. When enough buffer space is available, the width of the oscillations is bounded by the product of the bottleneck rate by the maximum delay of the SSE domain in front of the BNQ: $\Delta Q_{Max} = r \cdot (\delta + h) = 11\,MB$ in the $r = 8\,Gbps$ case. The reason for these short-term oscillations is easy to find: when the SSE domain is in the sleep state (for a total time $\delta + h$), the input queue of the SSE domain accumulates packets at a rate ρ that is slightly higher than the bottleneck rate: $\rho = r + \varepsilon$. The r component is sustained by the drain rate of the BNQ. The ε component results from the continuous increase of source activity in the TCP connections (at the pace of one maximum segment size per RTT for sources in congestion avoidance state). When the SSE domain in front of the BNQ returns to the active state, its queue backlog moves to the BNQ at a rate $C - r$ ($2\,Gbps$ in the example of Figure 15). The cycle repeats when the SSE domain in front of the BNQ returns to the sleep state the first time its input queue becomes empty again. Every cycle results in a short-term queue length oscillation of width bounded by ΔQ_{Max} and in a small BNQ length increase Δq induced by the ε component of ρ. Over a much longer timescale, which primarily depends on the number of TCP flows and on the distribution of their RTT values, the accumulation of the Δq contributions translates into queue congestion conditions.

The short-term queue length oscillations are not a sign of ongoing congestion and should have no role in its detection. To mask out the contribution of the oscillations to the IQL and AQL variables that drive the PED operation, we increase the IQL thresholds and the total buffer size by $\Delta Q_{Max} = r \cdot (\delta + h)$ and the AQL thresholds by $\Delta Q_{Max} / 2$. The effects of the refinement can be appreciated in the plot of Figure 16, where the queue is never empty despite short-term oscillations that are almost $11\,MB$ wide.

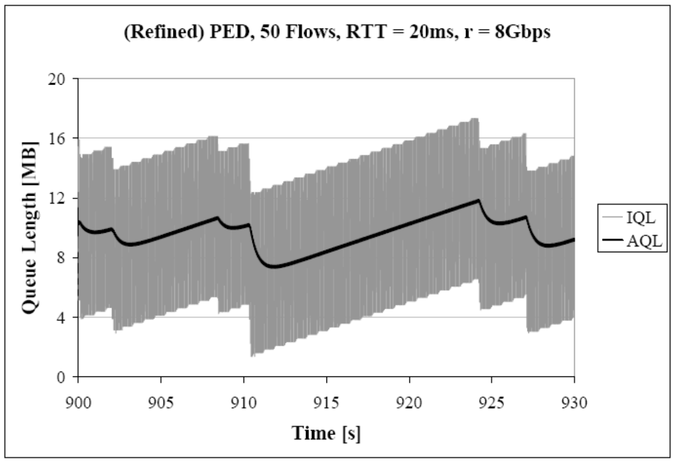

Figure 16. BNQ length evolution with refined configuration of the PED thresholds (30s interval).

Overall, we can draw the following conclusions from our experiments with SSE and PED: (a) The presence of SSE domains requires the adjustment of the PED thresholds for the bottleneck queue. Most important for engineering purposes is the increase by ΔQ_{Max} of the total buffer size. (b) Since the configuration of the PED thresholds is generally independent of the RTT distribution [7], the number of the SSE domains and the total added delays that they procure to the forward and return paths do not have immediate impact on the selection of the refined PED thresholds. (c) The extent of the required PED threshold increases is determined by the timing parameters (δ and h) of the SSE domains. Conservatively, to cover the whole range of bottleneck rates possible on a link with capacity C, the buffer size correction should be $\Delta Q_{Max} = C \cdot (\delta + h)$. (d) When the SSE domains in the data path have different timing parameters, the domain with the highest value of $\delta + h$ defines the buffer size correction ΔQ_{Max}, independently of its placement within the entire domain sequence.

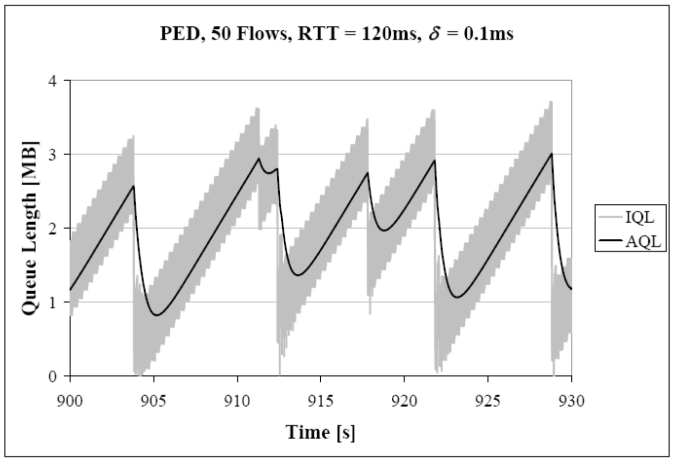

Figure 17. BNQ behavior with reduced transition times in the SSE domains, compensated by a 100ms extension of the RTT (30s interval).

Just like in the tail-drop case, in this section we have worked with relatively large values of δ and h with the intent of simplifying the recognition of the means of interaction between SSE and TCP. In more realistic scenarios with δ in the microsecond range, the relative impact of the buffer size correction becomes marginal compared to the original buffer size. For example, with $\delta = 0.1ms$ and $h = 1ms$ the buffer size increase is $\Delta Q_{Max} = 1.375\,MB$, or 17% of the original buffer size ($8\,MB$). Figure 17 shows the BNQ behavior when the PED thresholds are adapted to these faster SSE domains. In the experiment of the plot the propagation delay of the access links on the forward path is extended by $100\,ms$ to underscore that the adjustment of the PED threshold depends exclusively on the width of the short-term oscillations induced by the SSE domains and not on the contribution of the domains to the RTT (despite the larger RTT, the width of the short-term oscillations in Figure 17 is about one order of magnitude smaller than in Figure 16).

Finally, we present in Figure 18 the energy-saving performance of the ten SSE domains with the power data of Table I, and with $\delta = 0.1ms$ and $h = 1ms$. We remark that all experimental points (TCP) are fully aligned with the SSE upper bound on energy consumption. This is due to the bursty nature of the traffic aggregate produced by a highly synchronized set of TCP flows (synchronization is typical of small sets of flows with similar RTT values, as is the case in the traffic scenario of the experiment [19]). Still, the energy-saving performance is far from discomforting, because the distance between the upper and lower bounds is always very narrow for SSE.

We highlight the following key difference between tail drop and PED in defining the impact of SSE on TCP traffic: (a) With tail drop, the buffer size increase needed at the bottleneck link to compensate for the presence of the SSE domains is proportional to the total delay added by the SSE domains to the data path. (b) With PED, the buffer size increase is proportional to the delay added by a single SSE domain (if the added delay is not the same for all SSE domains, the largest delay dominates). These conclusions confirm the effectiveness of PED in supporting TCP traffic with minimal buffer resources.

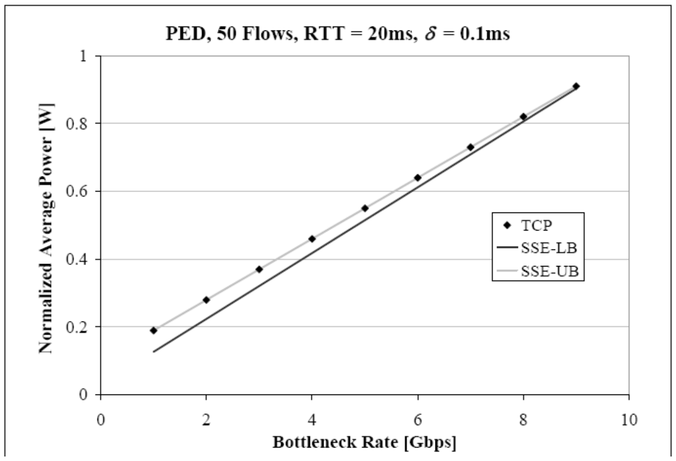

Figure 18. Average power consumption in the line of ten SSE domains with TCP traffic and PED buffer management.

V. Conclusions

Packet-timescale rate adaptation (PTRA) is a straightforward method for saving energy in the data-path hardware components of network elements. Its benefits can be substantial in every portion of the packet network, but most of all where traffic is sporadic and bursty. Unfortunately, the deployment of PTRA in commercial equipment is being delayed by the lack of a preferred approach for its implementation and by the uncertainty about the effects that PTRA may have on TCP traffic.

In this paper we presented conclusive solutions for both issues. First, we looked at the top candidates for introduction of the technology in commercial network equipment and concluded that sleep-state exploitation (SSE) should be the scheme of choice. The conclusion is based on the comparison of the performance of the candidate schemes under common deployment scenarios, where the trade-off between added delay and energy savings plays largely in favor of SSE versus its primary competitors. Next, we studied the impact of the presence of SSE domains along the data path of a set of TCP connections on the utilization of a bottleneck link. We observed that the buffer management policy in use at the bottleneck link modulates the extent of such impact. With the conventional tail-drop policy, the buffer size at the bottleneck link must increase proportionally to the overall delay added by all the SSE stages in the end-to-end path. Instead, with the recently defined periodic early detection (PED) scheme, the necessary buffer size increase is only proportional to the delay added by the "slowest" SSE stage in the path.

Overall, state transition times in the microsecond range will be the key technological enabler for making the pervasive presence of SSE devices in network equipment transparent to the end-user experience of network applications.

References

[1] L.A. Barroso and U. Holze, "The Case for Energy-Proportional Computing," IEEE Computer, 40:12 (2007), 33–37.

[2] S. Antonakopoulos, S. Fortune, L. Zhang, "Power-aware Routing with Rate-adaptive Network Elements," 3rd International Workshop on Green Communications (IEEE GreenComm3), Miami (FL), December 2010.

[3] D. Rossi, A.P. Bianzino, J.-L. Rougier, C. Chaudet, F. Larroca, "Energy-Aware Routing: a Reality Check," 3rd International Workshop on Green Communications (IEEE GreenComm3), Miami (FL), December 2010.

[4] L. Benini, P. Siegel, G. De Micheli, "Saving power by synthesizing gated clocks for sequential circuits," IEEE Design & Test of Computers, Volume 11, Issue 4, 1994.

[5] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing Network Energy Consumption via Sleeping and Rate Adaptation," Proc. of 5th USENIX Symposium on Networked Systems Design and Implementation (San Francisco, CA, 2008), pp. 323–336.

[6] A. Francini, D. Stiliadis, "Performance bounds of rate-adaptation schemes for energy-efficient routers," Proc. of IEEE HPSR 2010.

[7] A. Francini, "Beyond RED: Periodic Early Detection for On-Chip Buffer Memories in Network Elements," under submission, 2011.

[8] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," IEEE Journal of Solid-State Circuits, Vol. 35, No. 11, November 2000.

[9] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "The Limit of Dynamic Voltage Scaling and Insomniac Dynamic Voltage Scaling," IEEE Transactions on VLSI Systems, Vol. 13, No. 11, November 2005.

[10] D.C. Snowdon, S. Ruocco, and G. Heiser, "Power Management and Dynamic Voltage Scaling: Myths and Facts," Proc. of Workshop on Power Aware Real-Time Computing (Jersey City, NJ, 2005).

[11] Intel Corporation, "Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor," <ftp://download.intel.com/design/network/papers/30117401.pdf>, March 2004.

[12] ns2. <http://nsnam.isi.edu/nsnam/index.php/Main_Page>.

[13] IEEE P802.3az Energy Efficient Ethernet Task Force <http://www.ieee802.org/3/az/>.

[14] M. Gupta, S. Singh, "Greening of the Internet," Proceedings of ACM SIGCOMM 2003, August 2003.

[15] C. Villamizar and C. Song, "High-Performance TCP in ANSNET," ACM SIGCOMM Comp. Communications Review, 24(5):45-60, 1994.

[16] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, 1(4):397-413, 1993.

[17] V. Jacobson, K. Nichols, and K. Poduri, "RED in a Different Light," <http://www.cnaf.infn.it/~ferrari/papers/ispn/red_light_9_30.pdf>, 1999.

[18] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management," <http://icir.org/floyd/papers/adaptiveRed.pdf>, 2001.

[19] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," Proceedings of ACM SIGCOMM 2004.

VI. APPENDIX

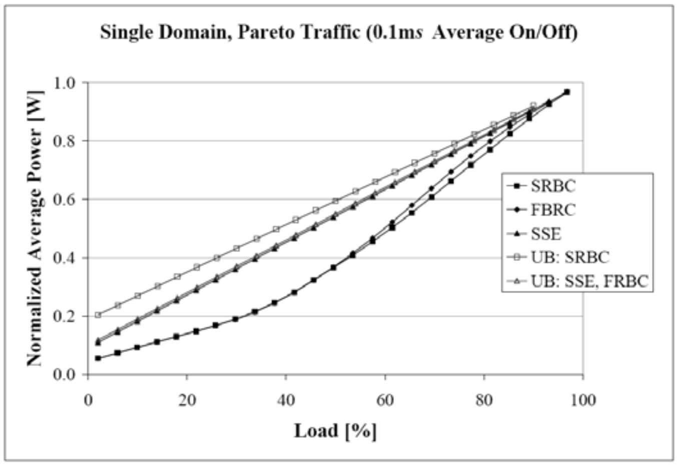

Figure A1. Average power consumption with Pareto traffic, compared to upper bounds (average on/off time smaller than the minimum hold time).

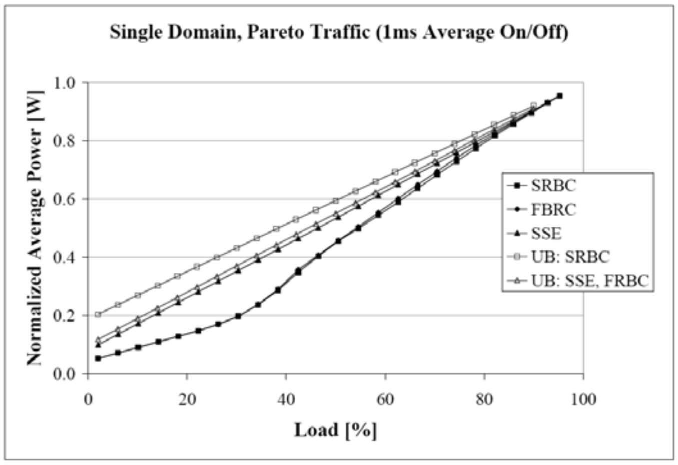

Figure A2. Average power consumption with Pareto traffic, compared to upper bounds (average on/off time as large as the minimum hold time).

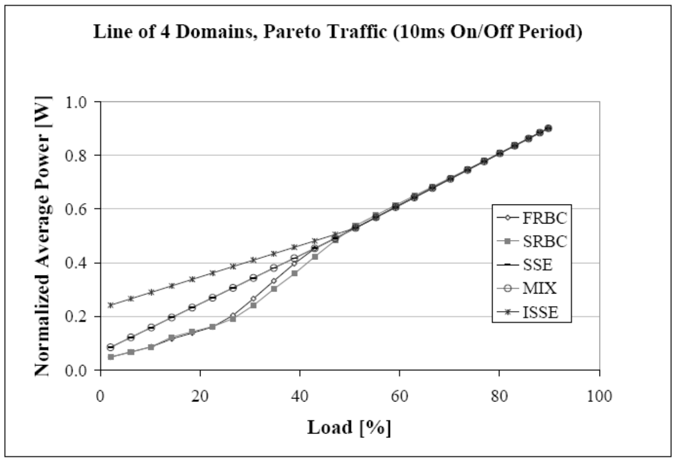

Figure A3. Average power consumption in a line of four PTRA domains (Pareto source with 10ms average on/off period).

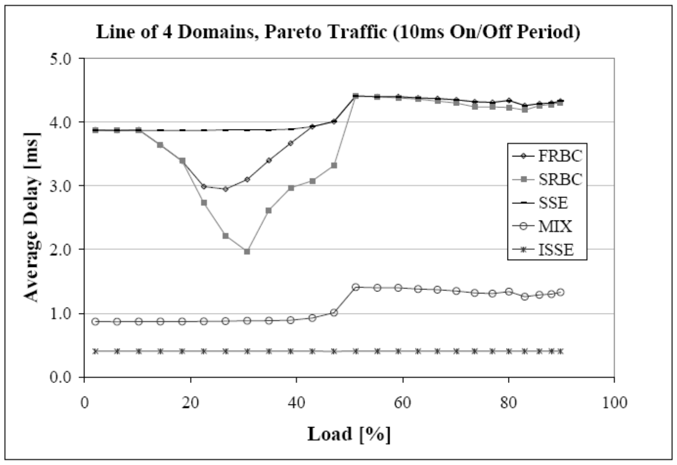

Figure A4. Average delay in a line of four PTRA domains (Pareto source with 10ms average on/off period).

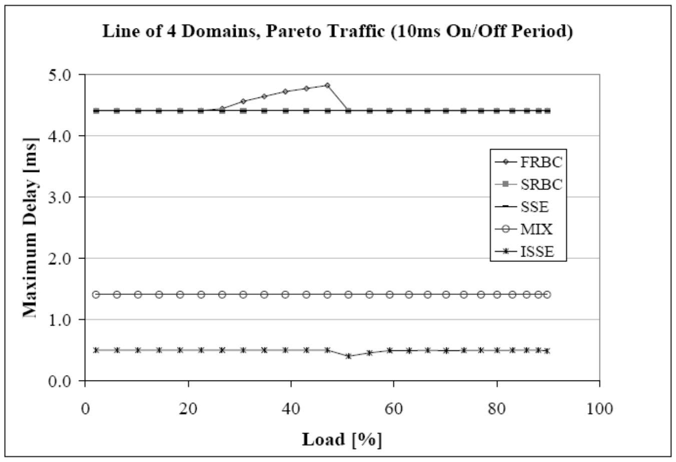

Figure A5. Maximum delay in a line of four PTRA domains (Pareto source with 10ms average on/off period).

Minimum-Cost Network Design with (Dis)economies of Scale

Matthew Andrews Spyridon Antonakopoulos Lisa Zhang

Bell Laboratories
600-700 Mountain Avenue
Murray Hill, NJ 07974
{andrews,spyros,ylz}@research.bell-labs.com

Abstract

Given a network, a set of demands and a cost function $f(\cdot)$, the min-cost network design problem is to route all demands with the objective of minimizing $\sum_e f(\ell_e)$, where ℓ_e is the total traffic load under the routing. We focus on cost functions of the form $f(x) = \sigma + x^\alpha$ for $x > 0$, with $f(0) = 0$. For $\alpha \leq 1$, $f(\cdot)$ is subadditive and exhibits behavior consistent with economies of scale. This problem corresponds to the well-studied Buy-at-Bulk network design problem and admits polylogarithmic approximation and hardness.

In this paper, we focus on the less studied scenario of $\alpha > 1$ with a positive startup cost $\sigma > 0$. Now, the cost function $f(\cdot)$ is neither subadditive nor superadditive. This is motivated by minimizing network-wide energy consumption when supporting a set of traffic demands. It is commonly accepted that, for some computing and communication devices, doubling processing speed more than doubles the energy consumption. Hence, in Economics parlance, such a cost function reflects *diseconomies of scale*.

We begin by discussing why existing routing techniques such as randomized rounding and tree-metric embedding fail to generalize directly. We then present our main contribution, which is a polylogarithmic approximation algorithm. We obtain this result by first deriving a bicriteria approximation for a related capacitated min-cost flow problem that we believe is interesting in its own right. Our approach for this problem builds upon the well-linked decomposition due to Chekuri-Khanna-Shepherd [12], the construction of expanders via matchings due to Khandekar-Rao-Vazirani [22], and edge-disjoint routing in well-connected graphs due to Rao-Zhou [29]. However, we also develop new techniques that allow us to keep a handle on the total cost, which was not a concern in the aforementioned literature.

1 Introduction

We consider a minimum-cost network design problem with the following familiar formulation. We are given a traffic matrix that specifies demands to be transported over a network. We also have a set of network resources that incur cost for carrying traffic. Specifically, each resource e is associated with a cost function $f_e(\cdot)$ such that a cost of $f_e(\ell_e)$ is incurred if e carries a traffic load of ℓ_e. The objective of the design problem is to choose a route for each traffic demand so that the total cost $\sum_e f_e(\ell_e)$ is minimized.

Buy-at-Bulk network design is a well-studied problem that falls under this formulation. For Buy-at-Bulk, the cost functions $f_e(\cdot)$ exhibit *economies of scale*. That is, higher traffic load yields lower cost per unit traffic carried. More precisely, the functions $f_e(\cdot)$ for Buy-at-Bulk are *subadditive*, i.e. $f_e(x) + f_e(x') \geq f_e(x + x')$ for any $x, x' \geq 0$. Buy-at-bulk has been extensively studied, because subadditive functions often model accurately the cost for purchasing link capacity in a variety of networks, whether it is a classic commodity network or a modern communication infrastructure. Polylogarithmic upper and lower bounds on the approximability of the Buy-at-Bulk problem are known, see e.g. [3, 10, 11, 1].

In this paper, we focus on a less studied case in which the cost function exhibits *(dis)economies of scale*. Our primary motivation for studying (dis)economies of scale is to take into account the *energy* cost of running a network. Energy conservation is attracting increasing attention in the fields of computing and networking, both because of the rapidly increasing monetary costs of powering large networks and server farms, as well as a desire to decrease the environmental impact of these operations [30, 25]. In this context, $f_e(\cdot)$ models an energy curve, reflecting how much energy e consumes as a function of its processing speed x. Here, e stands for a generic networking or computing device such as a CPU, communication link, edge router, etc. We assume that devices have the capability of *speed scaling*, which refers to adjusting the processing speed according to the traffic load. Speed scaling is popular research topic, see e.g. [32, 26, 5, 9, 4, 21, 19, 27]. It is also a feature in some commercial products such as the Intel Pentium processors [13], standards like ADSL2 and ADSL2+ [17], and proposals to the IEEE 802.3az task forces [14, 28].

Extensive studies have suggested that the energy consumption of some devices may exhibit *diseconomies of scale* and thus be characterized by *superadditive* functions, i.e. $f_e(x) + f_e(x') \leq f_e(x + x')$ for any $x, x' \geq 0$. This implies, for instance, that doubling processing speed more than doubles the energy consumption, which is particularly true if increasing the speed of a device requires increasing the clock speed of a microprocessor. Many papers model the power requirements of a microprocessor as a polynomial function of the clock speed, such as $f_e(x) = \delta_e x^\alpha$, where δ_e and α are parameters associated with the device. While α has been usually assumed to be around 3 [7], it has been recently estimated to be much smaller. In particular its value is 1.11, 1.66, and 1.62 for the Intel PXA 270, a TCP offload engine, and the Pentium M 770, respectively [31].

Min-cost network design under superadditive functions was recently studied in [2]. However, in general the problem can be inapproximable. For example, if the cost function is given by $f_e(x) = 0$ for $0 \leq x \leq 1$ and $f_e(x) = x - 1$ for $x > 1$, then it was shown in [2] that finding a routing that incurs zero total cost is equivalent to solving the Edge-Disjoint-Paths (EDP) problem. Since EDP is NP-hard, it follows that achieving any finite approximation ratio for the above cost function is also NP-hard. Nevertheless, this is a rather unnatural function, especially in the context of energy curves, since the cost remains zero even for some non-zero speeds. Under cost functions $f_e(x) = \delta_e x^\alpha$, which are more natural for modeling energy consumption, [2] showed that a variant of randomized rounding achieves a constant approximation ratio, assuming that α is a constant.

On the other hand, for a more accurate energy curve, a non-negligible startup cost is unavoidable. In particular, we are interested in cost functions of the following form:

$$f_e(x) = \begin{cases} 0 & \text{for } x = 0 \\ \sigma_e + \delta_e x^\alpha & \text{for } x > 0 \end{cases}, \tag{1}$$

where $\sigma_e > 0$ may represent e.g. the cost required to keep a device active at an almost-idle state, or a fixed amount of energy needed to turn on the device. For example, σ_e includes the significant energy consumption due to leakage currents [13]. We remark that if $\alpha = 1$, we obtain one of the classic Buy-at-Bulk cost functions, which involves a startup cost plus a linear function. However, for the case in which $\alpha > 1$, $f_e(x)$ is obviously no longer subadditive, since the superadditive term x^α dominates for large values of x. We refer to this problem as min-cost network design with *(dis)economies* of scale, putting parentheses around the prefix "dis-" to stress that $f_e(x)$ exhibits behavior consistent with both economies of scale (for sufficiently small x only) *and* with diseconomies of scale (for large x only).

The difficulty in devising approximation algorithms under (dis)economies of scale comes from the fact that it is hard to know whether our routing should be aiming for more aggregation of demands, which would lower the cost coming from the σ_e terms, or more separation of demands, which would lower the cost coming from the $\delta_e x^\alpha$ terms. As we explain later, these aspects of the cost function mean that standard techniques such as tree metric embedding or randomized rounding cannot yield a satisfactory approximation, at least not in a straightforward way.

1.1 Model and Results

More formally, for min-cost network design with (dis)economies of scale, we are given a network, represented by an undirected graph $G = (V, E)$, and a set $\mathcal{D} = \{1, 2, \ldots, k\}$ of traffic demands, where the i^{th} demand $(1 \le i \le k)$ is associated with an unordered pair of terminals $(s_i, t_i) \in V \times V$ and an integer $\mathsf{dem}_i > 0$ indicating the requested bandwidth. We assume links represent the abstracted resources, and each link $e \in E$ is associated with a cost function $f_e(\cdot)$. Our goal is to route all demands in an unsplittable fashion with the objective of minimizing the total cost $\sum_{e \in E} f_e(\ell_e)$, where the *load* ℓ_e of link e equals the total amount of traffic routed through e.

We focus on the *uniform* version of the cost function (1); namely, $f_e(\cdot)$ differs only by a constant factor from link to link:

$$f_e(x) = \begin{cases} 0 & \text{for } x = 0 \\ c_e(\sigma + x^\alpha) & \text{for } x > 0 \end{cases}. \tag{2}$$

Our main result is a polylogarithmic approximation algorithm for $\alpha > 1$ and $\sigma > 0$, in which case $f_e(\cdot)$ is neither superadditive nor subadditive.

As a byproduct, we obtain a polylogarithmic approximation for the *capacitated network design problem.* The precise relationship between the two problems is demonstrated later in Lemma 2. In the capacitated network design problem, we are given an undirected graph (or multigraph without self-loops) $G^\circ = (V^\circ, E^\circ)$, where each link $e \in E^\circ$ has cost κ_e and all links have capacity q, as well as a set $\mathcal{D}^\circ = \{1, 2, \ldots, k^\circ\}$ of traffic demands. Again, the i^{th} demand $(1 \le i \le k^\circ)$ is associated with an unordered pair of terminals $(s_i^\circ, t_i^\circ) \in V^\circ \times V^\circ$ and an integer $\mathsf{dem}_i^\circ > 0$ indicating the requested bandwidth. The goal is to route all demands in an unsplittable fashion, as before, while ensuring that the sum of bandwidths of the demands routed through each link e does not exceed

q. Here, our objective is to minimize the total cost of links used in the routing, i.e. those with non-zero load. Moreover, in our approximate solution, we allow the capacity on each edge to be exceeded by a polylogarithmic factor.

1.2 Related Work

Under uniform subadditive cost functions, the well-studied Buy-at-Bulk problem has an $O(\log n)$-approximation [3], where $n = |V|$, via tree metric embeddings [6, 15]; it is also hard to approximate to within an $\Omega\big(\log^{1/4-\varepsilon} n\big)$ factor [1]. Under non-uniform subadditive cost functions, i.e. when $f_e(\cdot)$ may be unrelated over all e, Buy-at-Bulk has polylogarithmic approximation [11, 24] and is hard to approximate to within an $\Omega\big(\log^{1/2-\varepsilon} n\big)$ factor [1].

For the superadditive function $f_e(x) = \delta_e x^\alpha$, randomized rounding can lead to a constant approximation for unit demands [2]. On the contrary, for any $\alpha > 1$, there is a uniform cost function of the form (2) such that no $\Omega\big(\log^{1/4-\varepsilon} n\big)$ approximation is possible, even for unit demands [2]. This hardness result follows easily from the hardness of Buy-at-Bulk. It also indicates that, when $\alpha \geq 1$, the introduction of a startup cost $\sigma > 0$ in the cost function makes the problem intrinsically harder to optimize.

We stress that the capacitated network design problem we consider in this paper has some important differences from the Generalized Steiner Network problem for which Jain's iterative rounding algorithm provides a 2-approximation. In the latter problem, a set $\{r_{ij}\}$ of demands is given, and the goal is to find a minimum cost network such that there are r_{ij} disjoint paths between each node pair (i, j). In another — much more difficult — version, these paths need not be disjoint, but up to q_e of them may use edge e. This variant was studied Carr et al. [8], who showed how to obtain an approximation ratio dependent on the number of non-zero coefficients in each row of the integer programming formulation. Observe, however, that these problems are quite different from ours, since the connectivity requirements of each demand need to be satisfied *in isolation*. By contrast, in our problem we aim to route *all* the demands *simultaneously*, and so the capacity of each edge must be no less than the *total* amount of traffic that is routed through it.

To the best of our knowledge, there are no previous polylogarithmic approximation algorithms for either our minimum cost network design problem with (dis)economies of scale or our formulation of capacitated network design. Nevertheless, we remark that the math programming community has studied cutting plane techniques for our capacitated problem, see e.g. [16].

1.3 A Hard Example

We now give a simple example for which neither randomized rounding nor routing based on tree metric embeddings yields a polylogarithmic approximation in the presence of (dis)economies of scale. This example was already discussed in [2], but we present it again here for completeness. Consider a pair of terminals (s, t), m parallel edges between them, and the cost function $f(x) = m + x^2$ for $x > 0$. We also have m unit demands that need to be routed between s and t. The optimal integral solution for this problem buys $\sqrt{m}$ edges and routes $\sqrt{m}$ of the demands on each edge. The total cost is $\Theta\big(m^{3/2}\big)$.

With randomized rounding, we first solve the corresponding linear relaxation. The fractional optimal solution may buy a $1/m$ fraction of each edge and route a $1/m$ fraction of every demand through it. The total cost for this fractional routing is $m(\frac{1}{m})m + m = 2m$, which illustrates that the *integrality gap* of this relaxation is at least $\Omega(\sqrt{m})$. Moreover, if we apply randomized rounding by treating the fractional value assigned to each route as a probability distribution and choosing a route for every demand according to that distribution, then the probability that an edge is picked equals the probability that some demand picks it when all the demands choose a route uniformly at random. This probability is $1 - (1 - \frac{1}{m})^m \geq 1 - \frac{1}{e}$. Therefore, the expected cost of the rounded solution is at least $m^2(1 - \frac{1}{e})$, which is an $\Omega(\sqrt{m})$ factor more than the optimal integral cost.

In the tree metric embedding approach that was used for the uniform Buy-at-Bulk problem, we approximate the underlying network with a tree and then solve the problem on that tree, with the same cost function. However, since our example only has two nodes, the only possible tree on the two nodes consists of a single edge connecting them. Hence when we solve the problem on that tree, all m demands are routed on one edge. Under our cost function, this solution has cost $m + m^2$, which is again an $\Omega(\sqrt{m})$ factor more than the optimal integral cost.

2 Overview

Given that demands can request different bandwidths, we first partition them into multiple buckets, each of which contains demands whose bandwidths are within a factor 2 of each other. Demands in the same bucket are therefore (almost) uniform in size, and we treat each bucket separately.

We set a parameter $\mu = \sqrt[\alpha]{\sigma}$. For a bucket of demands with bandwidth $\geq \mu$ each, we use randomized rounding to route them and achieve a constant approximation, as proposed in [2].

To route a bucket of small demands where $\mathsf{dem}_i < \mu$, we begin by *aggregating* the demands and creating *superterminals*, each of which gathers a $\Theta(\mu)$ amount of traffic. To relate the costs of the aggregated and the original instances, we make use of a minimum Steiner forest defined on the original demands. Similar aggregation approaches have been explored before, e.g. in [20, 12]. We remark that the significance of aggregation becomes apparent for a filtering procedure later on.

Subsequently, we convert the aggregated instance into an instance of capacitated network design, with the same demand set and the same network, except that we replace every link e by a set of parallel links with capacity μ each. The cost of the i^{th} such parallel link is given by $f_e(i\mu) - f_e((i-1)\mu)$. This may be viewed as a discretization of the cost function; see Figure 1. Lemma 2 implies that an approximation for the capacitated instance can be transformed into an approximation for the aggregated instance. Therefore, most of the proof concentrates on capacitated network design.

For a capacitated instance where demand sizes are comparable to link capacities, we begin by obtaining a fractional optimal solution. Using that, we then decompose the network graph into a number of components, each of which is *well-cut-linked.* We say that a graph is well-cut-linked if it has no small cuts with a large number of terminals on both sides of the cut. Chekuri et al. [12] introduced this notion of well-linkedness, and showed that any fractional flow can always be decomposed into a set of disjoint and well-cut-linked components without losing too much of the demand. Henceforth, we focus on one component at a time.

One of the key steps in our analysis is manipulating the fractional flow from the well-linked decomposition. We ensure, via a filtering procedure, that each of the terminals has at least λ flow emanating from it, where λ has a carefully chosen inverse polylogarithmic value. The demand aggregation in the prior step ensures that a polylogarithmic fraction of the demands survive this filtering process. We then use the integrality theorem of minimum-cost flow to argue that there is a flow *of no greater cost* than the aforementioned fractional flow, such that the fraction of each link that is used is a multiple of λ. This in turn implies that there is an *integral* solution of cost no greater than $1/\lambda$ times the minimum cost flow. As a result, we have a technique for obtaining

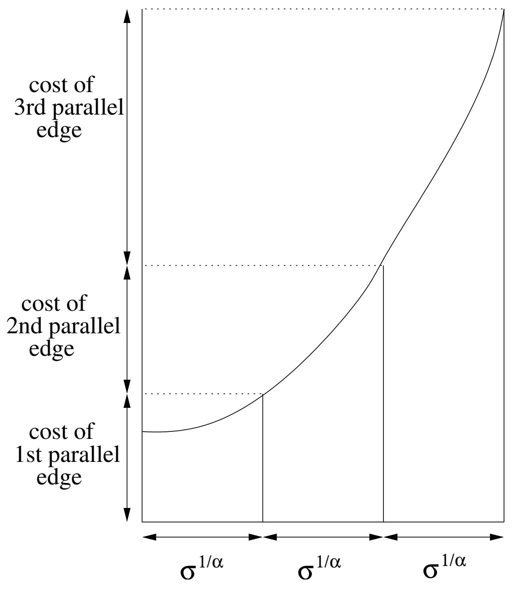
cost of 3rd parallel edge
cost of 2nd parallel edge
cost of 1st parallel edge
$\sigma^{1/\alpha}$
$\sigma^{1/\alpha}$
$\sigma^{1/\alpha}$

Figure 1: Discretization of the cost function.

cheap building blocks which, combined with the method of Khandekar-Rao-Vazirani [22], allows us to build an expander that can be embedded in the existing graph at low cost.

Using this expander, we would like to argue that we can route a polylogarithmic fraction of the demands in a disjoint manner, as in Rao-Zhou [29]. The catch here is that each superterminal collects $\Theta(\mu)$ traffic, which could prevent enough demands from simultaneously being routed disjointly. We therefore resort to a decomposition implied by König's Theorem [23] and route in $\Theta(\mu)$ rounds, where each round handles demands from distinct superterminals.

Not all demands are yet routed, as the filtering process and the disjoint routing within the expander only routes a polylogarithmic fraction of those. However, since we have an upper bound on the unrouted demands, it is straightforward to show that by recursively repeating the procedure a polylogarithmic number of times (and hence incurring a polylogarithmic factor in edge capacity violation and a polylogarithmic increase of the cost), we can in fact route all of them.

3 The algorithm

Before proceeding, we provide some additional definitions. In the following, $\mathcal{D}'$ stands for any given subset of the demands in $\mathcal{D}$. Denote the total demand $\sum_{i=1}^{k} \mathsf{dem}_i$ by D, the cost of the optimal solution by opt, and the minimum cost of a partial solution that routes the demands in $\mathcal{D}' \subseteq \mathcal{D}$ by $\mathsf{opt}_{\mathcal{D}'}$. Furthermore, let $\mu = \sqrt[\alpha]{\sigma}$ and $\mathsf{dmax} = \max_{i \in \mathcal{D}} \mathsf{dem}_i$.

Without loss of generality, we assume that each node is a terminal for at most one demand in $\mathcal{D}$. If that does not hold for some node $v \in V$, we simply create sufficiently many copies of v, each connected to v by an edge with zero cost coefficient, and replace v by a distinct copy of itself in whichever demand pair it appears. Clearly, this transformation does not affect the optimal solution cost, and the size of the transformed graph is only polynomially larger than that of the original one.

Procedure 1 Aggregation of small demands

```
initially all nodes in $V_p$ are unassigned
for $v \in V_p$ do
    $d(v) \leftarrow \mathsf{dem}_i$ if $\exists i \in \mathcal{D}_j$ s.t. $v = s_i$ or $v = t_i$, else 0
$w \leftarrow 0$; $W \leftarrow \emptyset$; $v_{\text{curr}} \leftarrow v_0$; $v_{\text{prev}} \leftarrow v_{\text{last}} \leftarrow$ node visited last in $T_p$
while $\sum_v d(v) \geq 3\mu$, summed over unassigned nodes do
    while $w < \mu$ do
        $w \leftarrow w + d(v_{\text{curr}})$; $W \leftarrow W \cup \{v_{\text{curr}}\}$
        $v_{\text{prev}} \leftarrow v_{\text{curr}}$; $v_{\text{curr}} \leftarrow$ node visited after $v_{\text{curr}}$ in $T_p$
    if $W \neq \emptyset$ then
        declare $v_{\text{prev}}$ a superterminal; assign all nodes in $W$ to $v_{\text{prev}}$
        $w \leftarrow 0$; $W \leftarrow \emptyset$
declare $v_{\text{last}}$ a superterminal; assign all currently unassinged nodes to $v_{\text{last}}$
```

3.1 Preprocessing

Bucketing demands. To begin with, partition $\mathcal{D}$ into $\zeta = \lfloor \log \mathsf{dmax} \rfloor + 1 = O(\log D)$ subsets. In particular, define $\mathcal{D}_j = \{ i \in \mathcal{D} \mid 2^{j-1} \leq \mathsf{dem}_i < 2^j \}$, $1 \leq j \leq \zeta$. Moreover, for every j and $i \in \mathcal{D}_j$, round dem_i up to 2^j; this adjustment entails only a constant factor loss in the approximation. In the following, we shall construct a partial solution for each $\mathcal{D}_j$.

Routing large demands. Note that if $2^j \geq \mu$, then in any partial solution that routes the demands in $\mathcal{D}_j$ only, for every network link $e \in E$ we have either $\ell_e = 0$ or $\ell_e \geq 2^j \geq \mu$. Consequently, we may approximate each function f_e by $f'_e(\ell_e) = 2c_e\ell_e^\alpha$, since $\frac{1}{2}f'_e(\ell_e) \leq f_e(\ell_e) \leq f'_e(\ell_e)$ for the aforementioned range of values of ℓ_e. Therefore, using the algorithm of Andrews et al. [2], we produce a partial solution that routes the demands in $\mathcal{D}_j$ with cost $\eta^\alpha \mathsf{opt}_{\mathcal{D}_j}$, where η is a constant.

Aggregating small demands. On the other hand, suppose that $2^j < \mu$. Let us construct an instance of the well-known Steiner forest problem on the graph G, with edge weights $c_e\sigma$ and terminal pairs (s_i, t_i), $i \in \mathcal{D}_j$. It is easy to see that the minimum weight of a Steiner forest is at most $\mathsf{opt}_{\mathcal{D}_j}$, hence by applying the 2-approximation algorithm in [18] we can find a Steiner forest H of weight $\leq 2\mathsf{opt}_{\mathcal{D}_j}$ efficiently.

Naturally, the connected components of H, say $H_1, \ldots, H_\theta$, are trees. Take each component $H_p = (V_p, E_p)$, $1 \leq p \leq \theta$, root it at an arbitrary leaf node $v_0 \in V_p$, and denote by T_p a depth-first-search traversal of H_p. Then, apply Procedure 1 on H_p to designate certain nodes of V_p as *superterminals* and assign each $v \in V_p$ to a superterminal.

For any $i \in \mathcal{D}_j$, let $\tilde{s}_i$ and $\tilde{t}_i$ be the superterminals to which s_i and t_i are assigned, respectively. For each demand $i \in \mathcal{D}_j$, create a so-called *aggregated demand*, simply by replacing the pair (s_i, t_i) with $(\tilde{s}_i, \tilde{t}_i)$. Then, consider an *aggregated instance* of the problem on the graph G with those aggregated demands only. We now have the following lemma.

Lemma 1. *Any solution to the aggregated instance specified above may be converted into a partial solution that routes the demands in $\mathcal{D}_j$, and vice versa. If C_{aggr} and C_{orig} are the respective costs of these solutions, then in one direction we can guarantee that $C_{\text{orig}} \leq 2^{\alpha-1}\big(C_{\text{aggr}} + 2(1+4^\alpha)\mathsf{opt}_{\mathcal{D}_j}\big)$, whereas in the other that $C_{\text{aggr}} \leq 2^{\alpha-1}\big(C_{\text{orig}} + 2(1+4^\alpha)\mathsf{opt}_{\mathcal{D}_j}\big)$.*

$$\begin{array}{lll}
\mathsf{LP3}: & \text{Fractional capacitated network design} & \\
\min & \displaystyle\sum_{e \in E^\circ} \kappa_e \sum_{i=1}^{k^\circ} x_{e,i} & \text{(3a)} \\
\text{subject to} & \displaystyle\sum_{i=1}^{k^\circ} x_{e,i} \le q \qquad \forall e \in E^\circ & \text{(3b)} \\
& \langle \text{flow conservation constraints on } x_{e,i} \rangle & \\
& 0 \le x_{e,i} \le 1 \qquad \forall e \in E^\circ,\ \forall i = 1,2,\ldots,k^\circ &
\end{array}$$

Sketch of proof. Using the edges of H, for $i \in \mathcal{D}_j$ we route dem_i flow between s_i and $\tilde{s}_i$, as well as between t_i and $\tilde{t}_i$. By the construction of superterminals, the load on any edge of H is $< \mu + 3\mu = 4\mu$ in this routing. Hence, its cost is bounded by $\sum_{e \in H}(1+4^\alpha)c_e\sigma \le 2(1+4^\alpha)\mathsf{opt}_{\mathcal{D}_j}$, and when combined with the aforementioned solution to the aggregated instance, it produces a partial solution for the demands in $\mathcal{D}_j$ with cost not exceeding $2^{\alpha-1}\big(C_{\text{aggr}} + 2(1+4^\alpha)\mathsf{opt}_{\mathcal{D}_j}\big)$. Finally, the argument for the opposite direction is entirely similar. $\square$

Reduction to capacitated network design. At this point, let us create an instance of the capacitated min-cost network design problem. The multigraph $G^\circ = (V^\circ, E^\circ)$ has the same node set $V^\circ = V$ as G. Moreover, for every link $e \in E$, we add at most $\omega = |\mathcal{D}_j| \le k$ parallel edges $e_1, e_2, \ldots, e_\omega$ to E°, where e_z has cost $c_e\sigma(z^\alpha - (z-1)^\alpha)$, $1 \le z \le \omega$. The edge capacity q is set to μ, and the demand set $\mathcal{D}^\circ$ consists of exactly the same (aggregated) demands as in the aggregated instance discussed earlier. Owing to our choice of edge costs, the next lemma is readily verified.

Lemma 2. *The ratio of the optimal cost of the capacitated min-cost network design instance defined above to the optimal cost of the aggregated instance lies between* 1 *and* 2.

3.2 Solving capacitated network design

A fractional relaxation of the capacitated network design problem can be formulated as the linear program $\mathsf{LP3}$, where the variable $x_{e,i}$ indicates the fraction of demand $i \in \mathcal{D}^\circ$ that is routed along link $e \in E^\circ$. Provided that $\mathsf{LP3}$ is feasible, let $\mathsf{opt}_{\mathsf{LP3}}$ be the optimal (fractional) solution cost. Our objective is to find an integral routing, i.e. one in which all $x_{e,i} \in \{0,1\}$, such that the total cost of edges used is at most $\beta\mathsf{opt}_{\mathsf{LP3}}$ and the load on each edge is at most γq — where β and γ are specified later.

3.2.1 The case $q = 1$

In this section we describe a solution to the special case of unit capacity ($q = 1$) and generalize it to $q > 1$ in Section 3.2.2. For $q = 1$, dem_i° should be 1 for all $i \in \mathcal{D}^\circ$. Demand aggregation in the preprocessing step is therefore not necessary. This allows us to continue to assume without loss of generality that the demand terminals are distinct. As we shall see in step 4 of the algorithm, distinct terminals make edge-disjoint routing in expander graphs more manageable.

Step 1. We obtain a fractional optimal solution $\mathbf{x}_{i,e}$ for the linear program $\mathsf{LP3}$. Let $\tau_e = \sum_i \mathbf{x}_{i,e}$ be the load on edge e in the fractional solution. Take G^τ to be the same graph as G°, but with each edge e having capacity τ_e instead of q. Clearly, the fractional routing implied by $\mathbf{x}_{i,e}$ is still feasible in G^τ. Moreover, $\mathsf{opt}_{\mathsf{LP3}} = \sum_e \kappa_e \tau_e$ is a lower bound on the cost of any integral routing in G° that respects edge capacities.

Step 2. We apply the following theorem of Chekuri, Khanna and Shepherd [12].

Theorem 3 ([12]). *We can decompose G^τ into node-disjoint subgraphs $G^\tau_1, G^\tau_2, \ldots, G^\tau_\phi$ and produce a weight function π on the terminals with the properties listed below. Denote by $\mathcal{D}^\circ_r \subseteq \mathcal{D}^\circ$ the set of induced terminal pairs in G^τ_r, and by X_r the set of terminals in $\mathcal{D}^\circ_r$.*

- $0 \leq \pi(s_i) = \pi(t_i) \leq 1$ *for all* $i \in \mathcal{D}^\circ$.
- *Each G^τ_r is* π-cut-linked. *That is, for any $A \subseteq V(G^\tau_r)$, the capacity on the cut defined by A and $\bar{A} = V(G^\tau_r) \setminus A$ in G^τ_r is at least* $\min\{\pi(A), \pi(\bar{A})\}$.
- $\sum_{r=1}^{\phi} \pi(X_r) = \Omega\big(|\mathcal{D}^\circ| \log^{-2} n^\circ\big)$, *where n° is the number of nodes in G°.*

Let $\lambda = 1 \big/ \lceil \log^3 n^\circ \rceil$. We define a new function ρ on the terminals, which is closely related to π. For a terminal $u \in X_r$, let

$$\rho(u) = \begin{cases} 0 & \text{if } \pi(u) < \lambda; \\ \lambda & \text{otherwise.} \end{cases}$$

Lemma 4. *The function ρ has properties similar to those of π. More specifically, (a) $0 \leq \rho(s_i) = \rho(t_i) \leq 1$ for all $i \in \mathcal{D}^\circ$; (b) each G^τ_r is* ρ-cut-linked*; and (c) $\sum_{r=1}^{\phi} \rho(X_r) = \Omega\big(|\mathcal{D}^\circ| \lambda \log^{-2} n^\circ\big)$.*

Proof. The first property follows directly from the fact that $\pi(s_i) = \pi(t_i)$ and the definition of ρ.

To verify the second property, consider any $A \subseteq V(G^\tau_r)$ and $\bar{A} = V(G^\tau_r) \setminus A$. Since G^τ_r is π-cut-linked, the capacity of the cut defined by A and $\bar{A}$ is at least $\min\{\pi(A), \pi(\bar{A})\}$. By the definition of ρ, $\pi(A) \geq \rho(A)$ and $\pi(\bar{A}) \geq \rho(\bar{A})$. Consequently, $\min\{\rho(A), \rho(\bar{A})\} \leq \min\{\pi(A), \pi(\bar{A})\}$, and hence the cut is at least $\min\{\rho(A), \rho(\bar{A})\}$.

For the last property, observe that $\sum_r \rho(X_r) \geq \lambda \left(\sum_r \pi(X_r) - \lambda |\mathcal{D}^\circ|\right)$, since the sum of π values that were reduced to zero in ρ is at most $\lambda|\mathcal{D}^\circ|$ and the remaining π values were reduced by at most a factor of λ. ☐

Step 3. Henceforth, we concentrate on one subgraph G^τ_r at a time. Recall that X_r is the set of terminals in G^τ_r, and let $X'_r = \{u \in X_r \mid \rho(u) = \lambda\}$. Since G^τ_r is ρ-cut-linked, we deduce that for *any* partition of X'_r into two equal halves (A, B), we can support a flow $\xi_{G^\tau_r}$ such that the amount of flow emanating from each node in A and the amount of flow absorbed by each node in B are both exactly λ. The existence of $\xi_{G^\tau_r}$ follows from the max-flow/min-cut theorem. Moreover, if $\xi_{G^\tau_r}(e)$ indicates the amount of flow on $e \in E(G^\tau_r)$, we have $\xi_{G^\tau_r}(e) \leq \tau_e$.

Now, let G^X_r have the same sets of nodes and edges as G^τ_r, but with capacity 1 for each edge. In G^X_r we determine a min-cost flow $\xi_{G^X_r}$ such that, again, the amount of flow emanating from each node in A and the amount of flow absorbed by each node in B are both exactly λ. Since 1 is an exact multiple of λ, the integrality theorem for min-cost flow guarantees that $\xi_{G^X_r}$ is λ-integral, meaning that $\xi_{G^X_r}(e)$ is an exact multiple of λ, for every $e \in E(G^X_r)$. Furthermore, $\sum_{e \in E(G^X_r)} \kappa_e \xi_{G^X_r}(e) \leq \sum_{e \in E(G^\tau_r)} \kappa_e \xi_{G^\tau_r}(e)$, because capacity constraints in G^X_r are more relaxed than those in G^τ_r.

Scaling up $\xi_{G_r^X}$ by a factor $1/\lambda$, we obtain an integral flow $\xi'_{G_r^X}$, under which the the amount of flow emanating from each node in A and the amount of flow absorbed by each node in B are both exactly 1. Therefore, $\xi'_{G_r^X}$ may be decomposed into $|A|$ paths, each carrying one unit of flow from a (distinct) node in A to a (also distinct) node in B. We call this a *path-matching* between the sets A and B. Note that any edge $e \in E(G_r^X) = E(G_r^\tau)$ belongs to at most $1/\lambda$ paths, and that the total cost of edges used by these paths is bounded by

$$\sum_{e\in E(G_r^X)} \kappa_e \xi'_{G_r^X}(e) = \sum_{e\in E(G_r^X)} \kappa_e \xi_{G_r^X}(e)/\lambda \le \sum_{e\in E(G_r^\tau)} \kappa_e \xi_{G_r^\tau}(e)/\lambda \le \sum_{e\in E(G_r^\tau)} \kappa_e \tau_e/\lambda.$$

Lemma 5. *Consider any equal partition (A,B) of X'_r. We can compute a path-matching between A and B within G_r^τ such that each edge in $G_r^\tau \subseteq G^\tau$ is used by no more than $1/\lambda$ such paths, and the total cost of links used is at most $\sum_{e\in E(G_r^\tau)} \kappa_e\tau_e/\lambda$.*

Step 4. At this point, we will use our above method for constructing path-matchings as a building block for our routing, in conjunction with two results stated below, due to Khandekar-Rao-Vazirani [22] and Rao-Zhou [29].

Theorem 6 ([22]). *Given a set $\mathcal{V}$ of N nodes and a procedure that finds a matching for any specified equal partition (A,B) of $\mathcal{V}$, we may efficiently determine $\psi = \Theta(\log^2 N)$ such partitions $(A_1,B_1),(A_2,B_2),\ldots,(A_\psi,B_\psi)$, so that the union of their corresponding matches results in a graph with $\Theta(1)$ expansion.*

Theorem 7 ([29]). *Consider an expander graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$, as in Theorem 6, and a number of pairs $(\hat{s}_i,\hat{t}_i) \in \binom{\mathcal{V}}{2}$. If each node belongs to at most one pair, then we may connect at least a $\Omega(\log^{-2} N)$ fraction of these pairs using edge-disjoint paths in $\mathcal{G}$.*

Since a path-matching between two equal-size node sets can be viewed as a (conceptual) matching, we may use the procedure implied by Lemma 5 in Theorem 6 to produce an "expander", each of whose edges represents in effect a path in G_r^τ. The union of these paths forms a subgraph of G_r^τ with total edge cost not exceeding $O(\log^2 n^\circ/\lambda)\cdot\sum_{e\in E(G_r^\tau)}\kappa_e\tau_e$, and every edge is contained in no more than $O(\log^2 n^\circ/\lambda)$ paths. Theorem 7 suggests that we can route at least an $\Omega(\log^{-2} n^\circ)$ fraction of the $|X'_r|/2$ demands with terminals in X'_r, using each of the aforementioned paths at most once. Across all node-disjoint subgraphs $G_1^\tau,\ldots,G_\phi^\tau$, $\Omega(|\mathcal{D}^\circ|\log^{-4} n^\circ)$ demands may thus be routed. Hence, we can route *all* the demands in $\mathcal{D}^\circ$ by recursively applying this entire process $O(\log^5 n^\circ)$ times, because $|\mathcal{D}^\circ|\left(1-\Omega(\log^{-4} n^\circ)\right)^{O(\log^5 n^\circ)} < |\mathcal{D}^\circ|/n^\circ \le 1$. The total cost of this solution is bounded by $O(\log^7 n^\circ/\lambda)\cdot \mathsf{opt}_{\mathsf{LP3}}$, and the load on each link is at most $O(\log^7 n^\circ/\lambda)$.

Theorem 8. *We have found an integral routing such that the total cost of edges used is at most $\beta\mathsf{opt}_{\mathsf{LP3}}$ and the load on each edge is at most γ, with $\beta = \gamma = \mathrm{polylog}(n^\circ)$.*

3.2.2 The case $q = \mu$

We now generalize our algorithm for the case $q = 1$ to the case $q = \mu$. Recall that demand aggregation in the preprocessing step in Section 3.1 creates superterminals each of which terminates demands of size between μ and 3μ. This creates an additional complication that not all of the demand at one superterminal X is necessarily destined to the same superterminal Y. We highlight the necessary changes in our algorithm.

For the aggregated instance on the superterminals, we first obtain an optimal fractional solution as before. We again perform the Chekuri et al. decomposition to obtain a set of well-linked instances. After this step some of the superterminals may only have a small amount of aggregate demand left. We therefore filter out all of the superterminals whose aggregate demand is less than λ, where λ is redefined to be $\mu/\lceil\log^3 n^\circ\rceil$. The new λ value and the fact that each superterminal initially had $\Theta(\mu)$ aggregated demands allow us to show that at least a $\lambda\log^{-2} n^\circ$ fraction of aggregated demands are not filtered. For the unfiltered demands, we again use the integrality theorem of min-cost flow and well-linkedness to show that there is a low-cost integral matching between any equal-sized partition of the superterminals.

However, due to aggregation each superterminal is not unique to some aggregated demand. In order to apply Theorem 7 we note that demands within a single bucket size equal to 2^j for some fixed j. König's lemma [23] states that we can decompose any bipartite graph with edge degree at most $3\mu/2^j$ into $3\mu/2^j$ disjoint matchings. Hence we can decompose the problem on the superterminals into $3\mu/2^j$ separate problems such that in each separate problem each superterminal represents at most 1 demand. Then whenever we construct an expander, we look at the separate problems one-by-one and disjointly route the demands for each. Whenever a demand is routed it consumes 2^j of the capacity on each edge along its route. Hence in order to route all of the $3\mu/\delta$ separate problems, we need at most 3 copies of each edge and the cost at most triples.

Therefore, Theorem 8 extends to $q = \mu$.

3.2.3 Wrapping up

Going back to the original network design problem with (dis)economy of scale, recall that we process one bucket of demands at a time. For the j^{th} bucket, $1 \le j \le \zeta$, we have a partial solution of cost at most $\text{polylog}(n)\,\mathsf{opt}_{\mathcal{D}_j}$. Combining these partial solutions yields a routing with cost $\zeta^\alpha\text{polylog}(n)\,\mathsf{opt} = \text{polylog}(n, D)\,\mathsf{opt}$, assuming that α is a constant.

Theorem 9. *Uniform network design with (dis)economies of scale has a* $\text{polylog}(n, D)$ *approximation.*

4 Conclusion

In this paper, we have derived a (β, γ)-bicriteria approximation for the uniform capacitated problem, where β, polylogarithmic in value, gives the cost guarantee and γ, also polylogarithmic in value, bounds the blowup in link capacity. We have also shown that this approximation implies a $\text{poly}(\beta, \gamma)$ approximation for the uniform min-cost network design problem under the cost function (2). We have focused on this particular cost function as it provides a natural model when considering the energy cost of a network. However, other cost functions can directly benefit from the (β, γ)-bicriteria approximation as well. For example, if $f(\lceil x/\mu\rceil\mu) \le \nu_1 f(x)$ and $f(\gamma x) \le \nu_2\gamma f(x)$ for some parameter μ, then the cost function admits an $O(\beta\nu_1\nu_2)$ approximation for min-cost network design. If ν_1 and ν_2 are polylogarithmic in size, then the resulting approximation is again polylogarithmic.

Of course, the main open problem is how to handle the non-uniform versions of the capacitated problem and the min-cost problem with (dis)economies of scale, respectively. We leave them both as challenging future work.

Acknowledgment. The authors wish to thank Chandra Chekuri for insightful discussions.

References

[1] Matthew Andrews. Hardness of buy-at-bulk network design. In *Proceedings of the 45th Annual Symposium on Foundations of Computer Science*, pages 115 – 124, Rome, Italy, October 2004.

[2] Matthew Andrews, Antonio Fernandez Anta, Lisa Zhang, and Wenbo Zhao. Routing for energy minimization in the speed scaling model. Manuscript, 2009.

[3] Baruch Awerbuch and Yossi Azar. Buy-at-bulk network design. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 542 – 547, Miami Beach, FL, October 1997.

[4] Nikhil Bansal, Ho-Leung Chan, and Kirk Pruhs. Speed scaling with an arbitrary power function. In *SODA '09: Proceedings of the Nineteenth Annual ACM -SIAM Symposium on Discrete Algorithms*, pages 693–701, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[5] Nikhil Bansal, Tracy Kimbrel, and Kirk Pruhs. Speed scaling to manage energy and temperature. *J. ACM*, 54(1), 2007.

[6] Yair Bartal. On approximating arbitrary metrics by tree metrics. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161 – 168, Dallas, TX, May 1998.

[7] David Brooks, Pradip Bose, Stanley Schuster, Hans M. Jacobson, Prabhakar Kudva, Alper Buyuktosunoglu, John-David Wellman, Victor V. Zyuban, Manish Gupta, and Peter W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000.

[8] Robert D. Carr, Lisa Fleischer, Vitus J. Leung, and Cynthia A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. In *SODA*, pages 106–115, 2000.

[9] Ho-Leung Chan, Wun-Tat Chan, Tak Wah Lam, Lap-Kei Lee, Kin-Sum Mak, and Prudence W. H. Wong. Energy efficient online deadline scheduling. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 795–804. SIAM, 2007.

[10] Moses Charikar and Adriana Karagiozova. On non-uniform multicommodity buy-at-bulk network design. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 176–182. ACM, 2005.

[11] Chandra Chekuri, Mohammad Taghi Hajiaghayi, Guy Kortsarz, and Mohammad R. Salavatipour. Approximation algorithms for non-uniform buy-at-bulk network design. In *FOCS*, pages 677–686. IEEE Computer Society, 2006.

[12] Chandra Chekuri, Sanjeev Khanna, and Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, 2005.

[13] Enhanced Intel SpeedStep technology for the Intel Pentium M processor. Intel White Paper 301170-001, 2004.

[14] IEEE P802.3az Energy Efficient Ethernet. Task force public area. http://grouper.ieee.org/groups/802/3/az/public/index.html, 2008.

[15] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the 35th Annual ACM Symposium of Theory of Computing*, pages 448 – 455, San Diego, CA, 2003.

[16] Bernard Gendron, Teodor Gabriel Crainic, and Antonio Frangioni. Multicommodity capacitated network design, 1998.

[17] George Ginis. Low-power modes for adsl2 and adsl2+. White Paper, Broadband Communications Group, Texas Instruments, Jan. 2005.

[18] Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24:296–317, 1995.

[19] Chamara Gunaratne, Kenneth J. Christensen, Bruce Nordman, and Stephen Suen. Reducing the energy consumption of ethernet with adaptive link rate (alr). *IEEE Trans. Computers*, 57(4):448–461, 2008.

[20] Anupam Gupta, Amit Kumar, and Tim Roughgarden. Simpler and better approximations for network design. In *Proceedings of ACM STOC*, pages 365–372, 2003.

[21] Sandy Irani and Kirk Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

[22] Rohit Khandekar, Satish Rao, and Umesh Vazirani. Graph partitioning using single commodity flows. In *38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 385–390, 2006.

[23] Dénes Kőnig. Graphok és matrixok. *Matematikai és Fizikai Lapok*, 38:116–119, 1931.

[24] Guy Kortsarz and Zeev Nutov. Approximating some network design problems with node costs. In *APPROX*, pages 231–243, 2009.

[25] Patrick Kurp. Green computing. *Commun. ACM*, 51(10):11–13, 2008.

[26] Minming Li, Becky Jie Liu, and Frances F. Yao. Min-energy voltage allocation for tree-structured tasks. In Lusheng Wang, editor, *COCOON*, volume 3595 of *Lecture Notes in Computer Science*, pages 283–296. Springer, 2005.

[27] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In Jon Crowcroft and Michael Dahlin, editors, *NSDI*, pages 323–336. USENIX Association, 2008.

[28] Prachi Patel-Predd. Energy-efficient ethernet. *IEEE Spectrum*, page 13, May 2008.

[29] Satish Rao and Shuheng Zhou. Edge disjoint paths in moderately connected graphs. In *International Colloquium on Automata, Languages and Programming (ICALP 06)*, Venice, Italy, 2006.

[30] SMART2020: Enabling the low carbon economy in the information age. www.smart2020.org, 2008.

[31] Adam Wierman, Lachlan L. H. Andrew, and Ao Tang. Power-aware speed scaling in processor sharing systems. In *INFOCOM*, 2009.

[32] F. Frances Yao, Alan J. Demers, and Scott Shenker. A scheduling model for reduced CPU energy. In *FOCS*, pages 374–382, 1995.

Power-aware Routing with Rate-adaptive Network Elements

Spyridon Antonakopoulos
Bell Laboratories, Murray Hill NJ
Email: spyros@research.bell-labs.com

Steven Fortune
Bell Laboratories, Murray Hill NJ
Email: sjf@research.bell-labs.com

Lisa Zhang
Bell Laboratories, Murray Hill NJ
Email: ylz@research.bell-labs.com

***Abstract*—Current Internet service-provider networks are typically overprovisioned, with actual traffic through a network element much less than the capacity of the network element. However, current network element power use is largely independent of actual traffic. This presents an opportunity to reduce network power use. This opportunity can be exploited locally, by redesigning individual network elements to make them rate-adaptive, or globally, by power-aware traffic routing.**

Instantiating either approach requires significant engineering effort. We attempt to quantify, as realistically as possible, the power-savings opportunity that can be obtained using these two approaches, in isolation or together. In particular, we investigate whether power-aware routing provides additional benefit if network elements are rate-adaptive.

We use a fairly simple model of network power use, where power consumption is attributed to links. A link can be turned off, in which case power use is zero, or a fixed startup power is required to turn on the link, after which power use increases with traffic. A significant parameter is the startup power as a fraction of full-capacity power. Since it is difficult to estimate feasible values for this parameter, we investigate various scenarios as this parameter varies from 0 to 100%.

We demonstrate that the combination of rate-adaptivity and power-aware routing saves a significant fraction of network power, for a wide variety of network topologies, traffic loads, and startup values. If the startup value is 50% or more, then power-aware routing appears to be of significant additional benefit over rate-adaptivity alone; if the startup value is 25% or less, than power-aware routing has relatively small additional benefit.

I. Introduction

Current Internet backbone networks are provisioned with much more capacity than average traffic [1]. Two major reasons for this over-provisioning are the significant daily variation in traffic load [11] and the need for redundant capacity to handle network element or link failures. However, the electricity use of current network elements (routers and switches) appears to be largely constant, independent of actual traffic [4]. This presents an opportunity to reduce backbone electricity use by making it more sensitive to traffic load.

One method to achieve this goal is via local power management within each network element, by making the power use of the element *rate-adaptive*, i.e. dependent on the traffic through the element. For example, a router typically has a network processor that processes packets for IP address lookups, congestion control, and policy enforcement. Techniques that are well-known in the computer industry for power management (speed-scaling, clock gating, sleep modes, etc.) can be applied to network processors as well, and significant power savings are predicted to follow [7].

[1]This R&D work was completed with the support of the U.S. Department of Energy (DOE), award no. DE-EE0002887. However, any opinions, findings, conclusions and recommendations expressed herein are those of the authors and do not necessarily reflect the views of the DOE.

Power management can also take place globally, at the network management level. Traffic within the network may be routed in a way that minimizes the total power used by the network. Such traffic management assumes that the relationship between traffic and power consumption is well understood for all network elements, and that it is feasible to compute a routing that achieves substantial power savings.

Instantiating either of these approaches presents formidable challenges: the local case necessitates a redesign of network element hardware and software; the global case requires a management system that is capable of network-wide traffic rerouting in response to traffic fluctuations.

The goal of this paper is to obtain as realistic as possible a prediction of the power savings that could be obtained by using one or both of these approaches. Both approaches address the same opportunity, so a specific question is whether power-aware routing is still useful even if network elements have been redesigned to be rate-adaptive. We estimate power consumption by developing models for the power requirements of network elements, and then simulating various network topologies under varying traffic loads.

The power use of a typical router can be attributed to packet-processing, to the switch fabric, and to the physical interface within every line card, with each of the above consuming 25-50% of the total. We do not yet know the full extent to which each function can be made rate-adaptive; packet-processing seems to be the easiest and the physical interface the most difficult to optimize.

We employ a relatively simple model in which total network power consumption is allocated on a per-link basis, with a fixed startup cost for activating the link and an additional cost that is dependent on traffic. (The startup cost is in effect the complement of the ‘energy proportionality index’ (EPI) [8]—expressed in percent, startup plus EPI sum to 100.) Future feasible values for the startup cost depend upon the detailed engineering of a network element and hence are difficult to estimate. We consider several possible values for the startup cost, ranging from zero to 100%. We do assume that a link may be turned off completely, using no power at all.

This study is explicitly scale-invariant: we predict relative but not absolute power savings. Some estimates on router power consumption are available [4], [12], which together with absolute traffic estimates can predict absolute power savings.

Traffic engineering in service-provider networks has many competing objectives and constraints, e.g. delay minimization, quality-of-service guarantees, reliability, business agreements, etc. It is beyond the scope of this paper to study these constraints in combination with power minimization. We instead attempt to estimate the power-saving opportunity. Exploiting this opportunity has many significant technical challenges, for example, an equipment failure make require rapid reactivation of nonlocal links that have been turned off to save power.

Section II below describes how we generated test instances, Section III the experiments, Section IV the corresponding results, and Section V the conclusion.

Other work. Power-aware routing has received considerable attention, including [4], [8], [9], [10]. We believe that we are the first to study systematically the combined effects of power-aware routing and rate-adaptivity with a startup cost.

II. Test-instance Methodology

A test instance has a network topology with capacitated links, a traffic matrix T, and for each link e, a power-traffic function $f_e(x)$ that determines the power needed to carry x units of traffic on e. We describe the methodology we used to generate synthetic test instances. The goal was to obtain test instances that model correctly-provisioned service provider networks; to do this we mimic the process by which a service provider might design its network. The instance-generation methodology only assumes a network topology and infers from it the traffic matrix, link capacities, and power-traffic curves.

The network topologies in our test instances are derived from the datasets in the Rocketfuel study [2]. This study used packet-tracing technology to infer the router-level topologies of autonomous systems in Europe, Australia and the US. From a Rocketfuel dataset we derive two network topologies. The first one, the *router graph*, is the router-level topology from the dataset, with minor cleanups (e.g. discarding nodes and links not in the largest connected component). The Rocketfuel dataset also specifies the city containing each router; typically there are multiple routers at each city. The second topology, the *city graph*, is obtained by collapsing all routers at a city together, and then merging together any resulting parallel links. The city and router graphs for Rocketfuel AS 3967 are shown in Figure 1. The router and city graphs are structurally similar; however the router graph has more links than the city graph, and hence there are more opportunities for a power-aware routing algorithm to turn links on and off.

The traffic matrix is based on a variant of the population-distance model, in which the traffic between two nodes is proportional to the product of their respective populations divided by a distance factor [6]. We estimate 'population' by the number of routers at the corresponding city (under the assumption that the service provider has approximately the right number of routers to handle the traffic generated by the

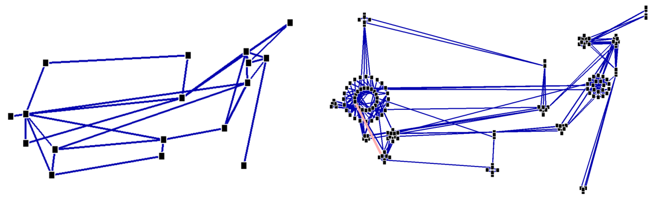

Fig. 1. City graph (left) and router graph (right) from the Rocketfuel AS 3967.

city). In particular, the traffic entry $T[a,b]$ is

$$T[a,b] = \frac{|\text{routers}(a)| \cdot |\text{routers}(b)|}{e^{\text{distance}(a,b)/1500}} \ .$$

Link capacities are chosen so that traffic can still be routed even if any single node fails. Such link capacities are easy to compute: we remove each node in turn from both network and traffic matrix, and route the remaining traffic in the network using shortest-hop routing. The capacity of each link is set to the maximum load on the link, over all choices of removed node.

The power-traffic function of each link e depends upon an instance-specific parameter β, the startup value, and the capacity of the link c_e. The power-traffic function linearly interpolates between βc_e for zero traffic and c_e for full traffic, but with a discontinuity at 0, where the function value is zero. Formally, the power-traffic function $f_e(x)$ for a link e with capacity c_e is

$$f_e(x) = \begin{cases} 0 & \text{if } x = 0 \ ; \\ \beta c_e + (1-\beta)x & \text{if } 0 < x \le c_e \ . \end{cases} \tag{1}$$

Notice that the marginal power per unit traffic is independent of the capacity of the link, but does depend upon β (in fact it is $1-\beta$). In Figure 2, plots 2–6 show five power curves for $\beta = 100\%, 75\%, 50\%, 25\%, 0\%$. The left-most plot is the "always-on" power-traffic function

$$f_e(x) = c_e \quad \text{if } 0 \le x \le c_e \ , \tag{2}$$

which models the situation that power consumption is oblivious to fluctuations in traffic.

III. Experiments

For each test instance, we wished to determine the relative benefits of rate adaptivity and power-aware routing. To do this, we computed a scaled traffic matrix T_α by multiplying every entry in T by α, for values of α in the range 10% to 100%. We then attempted to compute a power-optimal routing for each traffic matrix T_α. A generic form of the result is given in Figure 3. The top curve, always constant, shows power-traffic function (2) when routers are always on. The second curve, always linear, is the improvement due to rate adaptivity. The third curve shows the further improvement due to power-aware routing. The bottom curve is a straight line indicating perfect rate-adaptivity (i.e. 0% startup cost); this bounds the improvement possible by power-aware routing.

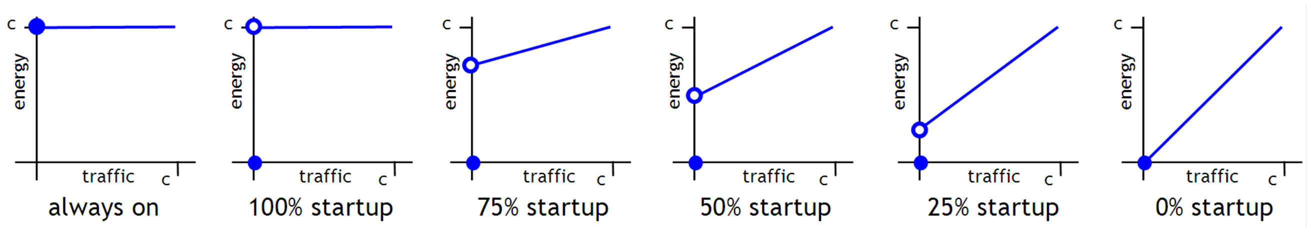

Fig. 2. The “always-on” power-traffic function, and linear power-traffic functions with various startup costs.

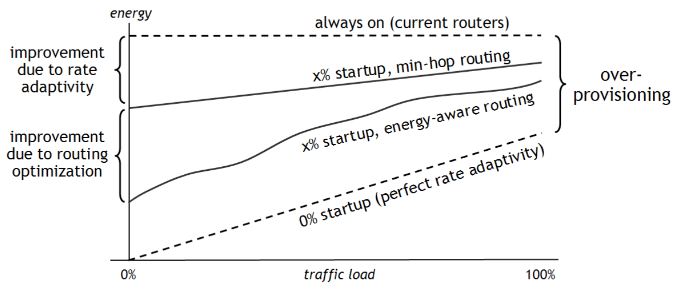

Fig. 3. Qualitative plot of power consumption using min-hop and power-aware routing, including the extreme cases of no rate adaptivity (always on) and perfect rate adaptivity.

Computing power-optimal routing is an NP-hard problem, and we did not attempt to find the globally optimal solution. Instead we used a heuristic, which might be called ‘iterative greedy least-power routing’. Initially the network does not carry any traffic and no links are turned on. Each demand is considered one by one, and routed on the path of least marginal cost, where ‘cost’ is the power required to the carry the demand. The marginal cost of a link is proportional to the size of the demand, plus the startup power of the link if the link has not been turned on. During the first pass, the route of each demand is computed although some demands may not be routable because of capacity constraints. For each subsequent pass, the route of each demand is recomputed assuming existing routes of other demands. The rerouting continues as long as more demands become routable or the total power incurred is reduced.

We cannot provide a worst-case guarantee on the quality of the solution provided by this routing heuristic. Its theoretical underpinnings stem from work by Charikar and Karagiozova [5]. However, it is simple and did substantially decrease network power when there was opportunity to do so.

IV. Simulation Results

We constructed test instances from multiple Rocketfuel datasets; to save space we only report on AS 3967 and AS 1755. The router versions of these two instances have about 150 nodes and 350 links; the city version of AS 3967 has 17 nodes and 27 links (Figure 1), while the city version of AS 1755 has 23 nodes and 38 links.

The power usage curves as function of traffic load for the city instance of AS 1755 are given in figure 4. The plot in the upper left corner (100% startup) assumes that there is no rate-adaptivity in a router, thus the benefit is solely from turning links off; in conditions of light load (30%), the benefit is substantial—roughly half the power. As the startup cost is decreased, the benefit of turning links off also decreases, so that in the lower right hand corner (25% startup) there is little additional benefit in turning links off.

This behavior was qualitatively similar over all the instances that we tested. Figure 5 shows two of the plots for the city instance of AS 3967. In this case, the relative benefit of turning links off was somewhat less, but still significant, particularly during light load.

For each AS, the router graph has similar structure to the city graph, but with more links. We expected that the extra links would create more opportunities for saving power by turning links off, and hence provide more opportunity for the power-aware routing algorithm. Figure 6 shows the behavior for the router graph of AS 3967. As expected, power-aware routing does slightly better than in the city-graph case (Figure 5), however the extra benefit is not large.

The test-instance methodology creates links with arbitrary capacities. Real network links only have a discrete choice of capacities (1Gbps, 10Gbps, 40Gbps, etc.). To see whether having discrete link capacities would change the results, we rounded link capacities up to the next highest integral power of 2. The results for AS 3967 are plotted in Figure 7. The upwards rounding substantially increases the initial over-provisioning of the network; however the relative behavior of rate-adaptation and power-aware routing was essentially unchanged.

Finally, we experimented with other models of the power-traffic function. Modern processors use dynamic voltage-frequency scaling to match processor clock rate to processing requirements; however power increases superlinearly with clock rate. We modeled this behavior with a convex power-traffic function with startup cost, specifically

$$f_e(x) = c_e(\beta + (1 - \beta)(x/c_e)^2)$$

adjusted to be zero at $x = 0$. Figure 8 shows the results, plotted simultaneously with the linear power-traffic curve. Network power in the convex case is predicted to be less than in the linear case (since this is true on a per-link basis); however the improvement due to routing is comparable in the two cases.

V. Conclusion

We believe that there is significant opportunity to save electricity in service-provider IP networks using rate-adaptivity

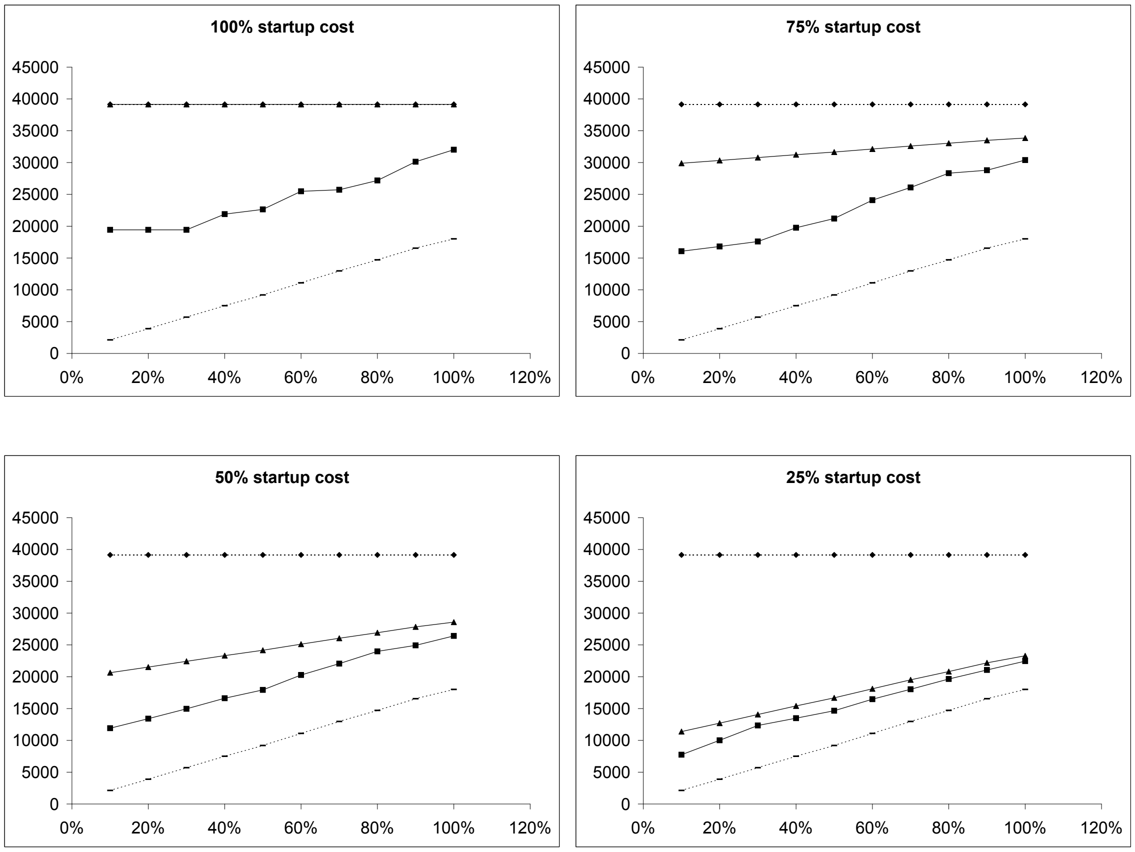

Fig. 4. Results for the city graph of AS 1755. In all plots, the x-axis indicates network load as a percentage, and the y-axis represents network power. Triangular (▲) and square (■) points denote results for minimum-hop and power-aware routing, respectively. Non-rate-adaptive (⧫) and perfectly rate-adaptive (–) are also plotted for reference.

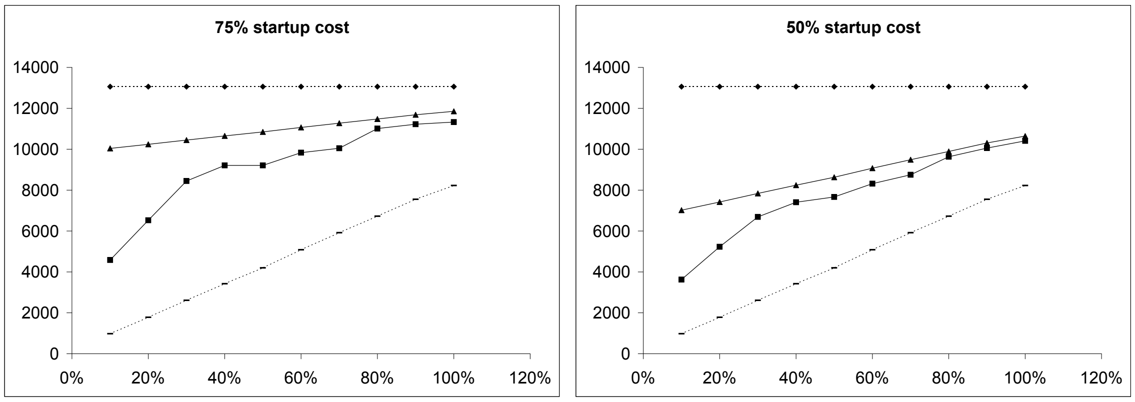

Fig. 5. Simulation results for the city graphs of Rocketfuel AS 3967.

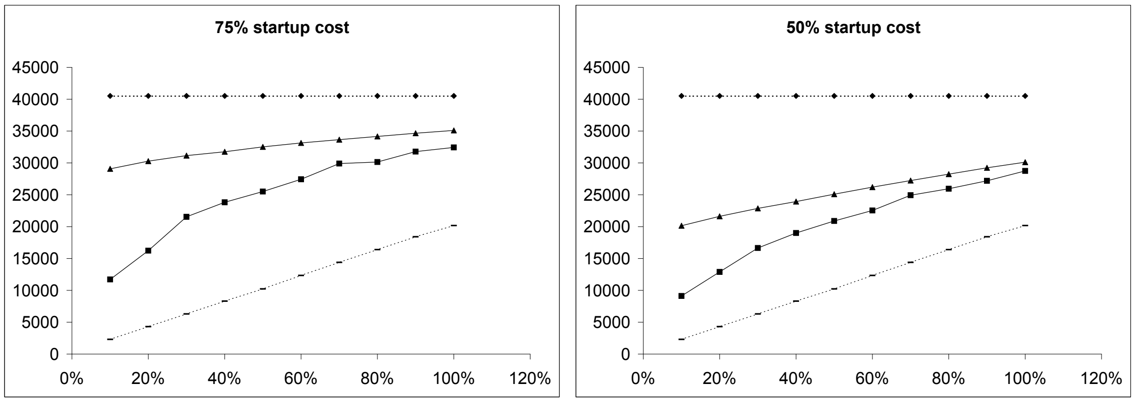

Fig. 6. Results for the router graph of AS 3967.

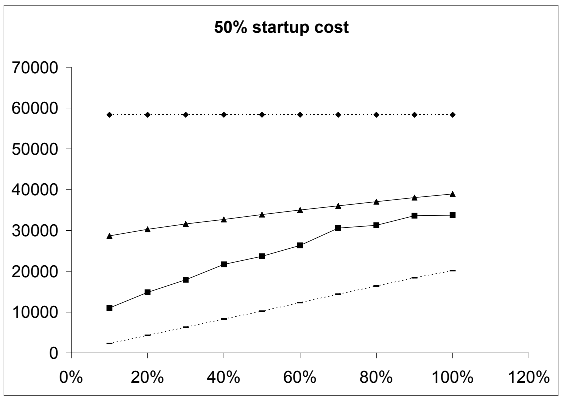

Fig. 7. The router graph of AS 3967 with quantized link capacities.

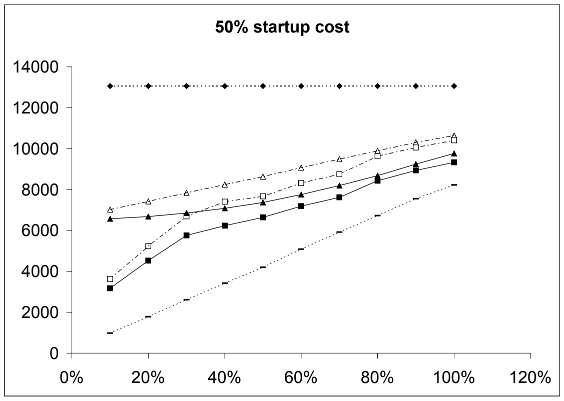

Fig. 8. The city graph of AS 3967 with the convex power curve (see text). Hollow plot points show the corresponding power costs for a linear power profile with the same startup cost.

and power-aware routing. Current network elements have poor rate adaptivity, with startup cost close to 100%. In this case, our results suggest significant benefit of power-aware routing, especially for low traffic. As rate adaptivity improves, the benefit of routing optimization diminishes. In the extreme case, if routers could be made perfectly rate-adaptive, then there would be no need for power-aware routing; unfortunately this hypothesis is unlikely.

Some graph-theoretic questions remain unanswered. For example, the power curves had some dependence on graph structure (Figure 4 versus Figure 5), a fact for which we do not have a good explanation. Such an explanation could lead to a criteria for choosing network graphs that are particularly power efficient.

References

[1] International Internet Statistics. TeleGeography Research, 2005. Available at http://www.itu.int/dms_pub/itu-d/md/02/isap2b.1.1/c/D02-ISAP2B.1.1-C-0025!!PDF-E.pdf.

[2] Rocketfuel: an ISP topology mapping engine. Available at http://www.cs.washington.edu/research/networking/rocketfuel/.

[3] Enhanced Intel Speedstep technology for the Intel Pentium M processor. Intel White Paper 301170-001, 2004.

[4] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, S. Wright, Power awareness in network design and routing, *INFOCOM 2008*.

[5] M. Charikar, A. Karagiozova. On non-uniform multicommodity buy-at-bulk network design. *Proc. of ACM STOC*, p. 176–182, 2005.

[6] S. Erlander, N. F. Stewart. The gravity model in transportation analysis - theory and extensions. *VSP* 1990.

[7] A. Francini, D. Stiliadis. Performance Bounds of Rate-Adaptation Schemes for Energy-Efficient Routers. *Proc. of IEEE HPSR*, 2010.

[8] P. Mahadevan, P. Sharma, S. Banerjee, P. Ranganathan, A Power Benchmarking Framework for Network Devices, *Proceedings of IFIP Networking Conference*, 2009.

[9] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, D. Wetherall, Reducing Network Energy Consumption via Sleeping and Rate-Adaptation, *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 323-336, April 2008.

[10] J. Restrepo, C. Gruber, C. Mas Machuca, Energy Profile Aware Routing, *First International Workshop on Green Communications IEEE International Conference on Communications (ICC)*, June 2009.

[11] M. Roughan, A. Greenber, C. Kalmanek, M. Rumescwicz, J. Yates, Y. Zhang. Experience in measuring Internet backbone traffic variability. *Proc. ACM SIGCOMM IMW*, p. 91–92, 2002.

[12] O. Tamm, C. Hermsmeyer, A.M. Rush, Eco-sustainable system and network architectures for future transport networks. *Bell Labs Technical Journal* 14: 311–327.

Beyond RED: Periodic Early Detection for On-Chip Buffer Memories in Network Elements

Andrea Francini
Alcatel-Lucent Bell Laboratories
Mooresville, NC (USA)
Email: andrea.francini@alcatel-lucent.com

Abstract—**The scalability and energy efficiency of future network equipment will critically depend on the ability to confine the memories that implement the packet buffers within the same traffic management chips that process and forward the packets. Despite massive research efforts aimed at trimming its demand of large buffers for the accommodation of TCP traffic, the bandwidth-delay product (BDP) rule remains to-date the dominant criterion for the sizing of packet buffers in commercial network elements, and arguably the only cause for their implementation in off-chip memories. Only the lack of a valid alternative justifies the lasting popularity of conventional buffer management methods for TCP traffic such as Tail Drop and Random Early Detection (RED), which fail to reconcile small buffer sizes with high-end throughput performance. Our contribution is twofold. First, we show that the RED algorithm is intrinsically flawed because of the way it maps buffer occupancy levels onto packet drop probabilities. Second, we introduce Periodic Early Detection (PED), a buffer management scheme with touchless configuration that sustains 100% link utilization using only 2.5% of the memory required by the BDP rule. While a more comprehensive study of PED's properties is in order, the clear superiority of the scheme under common benchmarking setups places it at the forefront of the candidate enablers for the on-chip implementation of buffer memories.**

I. INTRODUCTION

In typical routing and switching system designs, packet buffers hamper scalability and contribute prominently to energy consumption. Because of the size of the buffers, their instantiation commonly requires large memories that cannot be integrated in the same chips that process and forward the packets. As a consequence, memory chips and the connectivity infrastructure needed to reach them consume a substantial portion of real estate on circuit boards, taking it away from packet-handling devices. Off-chip memories also impose limitations on packet forwarding rates. Compared to the ideal case with on-chip memories only, external memories cause a substantial increase in area and power consumed per unit of forwarding capacity.

The temporary storage of IP packets generated by TCP sources is by far the primary purpose served by large packet buffers. Non-TCP traffic also needs buffering for the absorption of mismatches that may occur between arrival and departure rates in front of outgoing links. However, small packet buffers are sufficient to effectively resolve such mismatches, both short-term and long-term: in the former case by providing enough space for the accommodation of the excess packets, in the latter case by setting the conditions for their elimination, ideally after their accurate identification as offenders of respective pre-negotiated contracts [1]. Unfortunately, small buffers are not nearly as effective at handling TCP traffic.

For many years, it has been commonly accepted that, in front of a link of capacity C, a buffer space of size $C \cdot \overline{\vartheta}$ should be allocated for a queue that handles TCP traffic flows, where $\overline{\vartheta}$ is the average packet round-trip time (RTT) estimated over all the TCP flows in the queue. The goal of this bandwidth-delay product (BDP) buffer allocation rule, first advocated in [2], is to avoid queue underflow conditions, and therefore reductions of link utilization, as a consequence of packet losses occurring at times of traffic congestion. With $\overline{\vartheta} = 250\,ms$ and $C = 40\,Gbps$, which are typical values for 2010 core network links, the BDP buffer size is $1.25\,GB$. It is very unlikely that an embedded memory of this size can be implemented with current technologies. Moreover, $100\,Gbps$ links have started appearing in commercial applications, exacerbating even further the technology gap that faces the pervasive deployment of on-chip buffer memories (see [3] for a comprehensive discussion of the challenges involved in matching ever-growing link rates with adequate memory sizes).

The BDP rule assumes the application of a plain Tail Drop policy to the management of the buffer space: the buffer admits a newly arrived packet if it has enough space for it, otherwise it drops it. After observing that the BDP rule finds motivation in the worst-case assumption that the TCP flows traversing the bottleneck link are highly synchronized, the authors of [3] argue that the buffer size can safely be reduced in front of core network links, where *long-lived* TCP flows (i.e., flows that have reached at least once the congestion-avoidance phase) are numerous and not synchronized. Their *small buffer rule* recommends a Tail Drop buffer of size $Q_{max} = C \cdot \overline{\vartheta} / \sqrt{N}$ to hold the link utilization in proximity of 100%, where N is the number of long-lived TCP flows. The massive research effort that followed the publication of [3], thoroughly summarized in [4], not only fell short of yielding straightforward solutions for extending the results to more general settings with fewer and possibly synchronized long-lived flows, as in access and

This work was completed with the support of the U.S. Department of Energy (DOE), award no. DE-EE0002887. However, any opinions, findings, conclusions and recommendations expressed herein are those of the author and do not necessarily reflect the views of the DOE.

private-network links, but even led to a conservative revision of the small buffer rule when applied to core network links: $Q_{max} = C \cdot \overline{\vartheta} / 10$ [5].

A TCP source interprets a packet loss as an indication that congestion is occurring somewhere in the network, and reacts to it by reducing its traffic generation rate. For a given traffic mix at a bottleneck queue, the equilibrium that keeps the queue healthily loaded (neither empty nor overflowing) is the result of the right balance between the set of TCP flows in congestion avoidance state (slowly increasing their transmission rate) and the set of TCP flows that are recovering from recent packet losses. Such a balance is the product of an ideal packet drop rate $\varphi(t)$ that evolves over time with the composition of the traffic mix and with the sequence of packet loss assignments to individual TCP flows. We refer to $\varphi(t)$ as the *phantom drop rate*. We postulate that such a rate exists, but we never attempt to derive it analytically.

The Random Early Detection (RED) algorithm, first introduced in [6], remains to date the buffer management scheme that closest instantiates the idea of finding the equilibrium of the queue by proper approximation of its phantom drop rate. RED starts dropping packets far before the queue approaches saturation, allowing for time spacing between consecutive packet drop decisions. This way, RED notifies randomly selected TCP sources of their need to temporarily slow down in order to preserve the overall utilization of the congested link. A function that maps average queue length (AQL) levels onto packet drop probabilities statistically defines the spacing between packet drop decisions.

While RED has widely recognized merits and counts many implementations in commercial network equipment, its widespread use in practical networks is still hindered by the complexity of its configuration and the evident imperfection of its performance (both acknowledged and addressed, among others, by RED's original inventors [1], [7]). Among the many RED enhancements proposed in the literature, the Adaptive RED (ARED) scheme [7], [8] is one that fixes most configuration issues, leaving the network operator with only two parameters to choose: the target average queueing delay d and the estimated average round trip time $\overline{\vartheta}$. The analysis presented in [1] identifies and suitably solves a large number of performance issues, but fails to recognize the main reason for RED's incomplete success, just like any other enhanced-RED paper that we are aware of (see, for example, [7], [8], [9], [10]).

Simply put, RED as a control algorithm is intrinsically unstable. This is in plain contrast with the fundamental claim that "as long as its control law is monotone non-decreasing and covers the full range of 0 to 100% drop rate, RED works for any link, any bandwidth, any type of traffic" [11]. In this paper we show evidence that the source of RED's instability is indeed the monotonic function that maps AQL levels onto packet drop probabilities. With such a function, the packet drop probability can match the phantom drop rate only instantly, and not over extended time intervals. To ensure the stability of the queue length, the packet drop probability should instead remain constant for an extended period of time, at a value that closely approximates the phantom drop rate.

We rely on this basic observation to design a new buffer management scheme, called Periodic Early Detection (PED), which enforces fixed packet drop rates over extended time intervals. The choice of the packet drop rate is driven by a control algorithm that compares the AQL with a small set of thresholds. By decoupling the use of the AQL in the packet drop decisions and in the algorithm that controls the packet drop rate, we can afford much smaller queue length thresholds than those required by the BDP rule and those recommended in [1] for RED, fitting well into memory sizes that are compatible with the on-chip implementation of packet buffers.

We organize the paper as follows. In Section II we describe the RED algorithm in its basic formulation, discuss a series of RED enhancements that have been proposed in the literature, and provide evidence of RED's intrinsic instability. We specify the PED algorithm in Section III and show how it is configured in Section IV. In Section V we present simulation results for the appreciation of PED's performance. In Section VI we conclude the paper with a summary of its contributions and a plan for future work.

II. Random Early Detection

In this section we recall the native definition of the RED algorithm, review past proposals for improving it, and present evidence that the scheme, however enhanced, is unstable.

A. Native RED

According to the definition given in [6], upon arrival of the n-th packet a RED queue updates the AQL $\overline{q}$ based on the current value of the instantaneous queue length (IQL) q ($\overline{q}[n] = w \cdot q[n] + (1-w) \cdot q[n-1]$) and *marks* the packet with probability that depends on the current AQL value ($p[n] = p(\overline{q}[n])$). The marking of the packet has different consequences depending on the application of the algorithm. For convenience of presentation, in this paper we focus exclusively on the case where marking implies the immediate elimination of the packet. We expect the results of the paper to apply consistently to all applications of packet marking, such as marking for explicit congestion notification (ECN) [12].

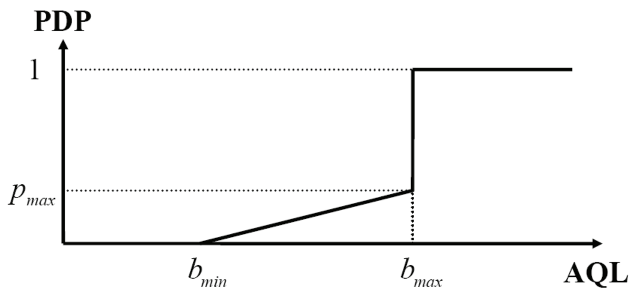

Figure 1. Drop probability curve of the native RED scheme.

Fig. 1 shows a typical profile of the function that maps AQL and packet drop probability (PDP) levels. With the function of Fig. 1, RED drops no packet as long as the AQL remains below the minimum threshold b_{min}. When the AQL is between the minimum threshold b_{min} and the maximum threshold b_{max}, an incoming packet is dropped with probability that depends linearly on the current position of the AQL between the two thresholds (the probability is p_{max} when $\overline{q} = b_{max}$). RED drops every incoming packet when $\overline{q} > b_{max}$ and also when the IQL exceeds the threshold $Q_{max} > b_{max}$ that defines the total buffer space available. The use of the AQL instead of the IQL isolates the packet drop decision from short-term IQL fluctuations that reflect ordinary TCP dynamics and not the onset of congestion conditions, and therefore should have no impact on the packet drop rate.

The operation of RED requires the configuration of the following parameters: the weight w of the exponential weighted moving average (EWMA) that computes the AQL, the buffer thresholds b_{min}, b_{max}, and Q_{max} (we assume that all thresholds and queue lengths are expressed in bytes), and the maximum drop probability p_{max}. Because of the large number of configuration dimensions involved, qualifiers like "black art" [1] and "inexact science" [13] have been widely attributed to the tuning of RED in its native formulation.

B. *Preferred RED Enhancements*

Soon after the IETF issued a "strong recommendation for testing, standardization, and widespread deployment of active queue management in routers," pointing to RED as a preferred candidate [14], the race started to better understand the scheme and eventually propose modifications that could strengthen its performance and simplify its configuration. Out of the vast body of RED enhancement proposals that were spawned by that race, here we focus on those that we consider instrumental to the correct identification of RED's core issue.

As observed in [1] and [10], the event-driven computation of the AQL, based exclusively on packet arrivals, is an obvious source of inaccuracy in the detection of congestion conditions. Instead, the AQL computation should be a time-driven process, with updates triggered by the expiration of a fixed averaging period τ_q that should be at least as large as the inter-departure time of packets of typical size (e.g., $1500\,bytes$) at the expected capacity of the bottleneck link (with $C = 40\,Gbps$, $\tau_q \geq 0.3\,\mu s$). Although the synchronization of the averaging process produces performance improvements that prove only marginal in most cases, highly beneficial is its impact on the configuration of the weight parameter of the EWMA. In fact, the fixed spacing of the AQL updates makes it possible to identify the EWMA with a discrete-time low-pass filter with time constant $T = \tau_q / w$. Since the purpose of the EWMA is the isolation of the AQL from IQL oscillations that occur at the RTT timescale, a time constant that is larger than the expected RTT for a large majority of the TCP flows handled by the queue is sufficient to smooth out undesired AQL fluctuations. With $T = 500\,ms$ (RTT values beyond $500\,ms$ are commonly considered unusual) and $\tau_q = 10\,\mu s$ (to reconcile the workload of the averaging process with the accuracy of the AQL computation), a weight $w = 0.00002$ works well for a $40\,Gbps$ link under most traffic scenarios (w scales with the inverse of the bottleneck rate). In this paper we only consider synchronous implementations of RED, with the choice of parameters that we have just described.

Important results have also been obtained in the choice of the maximum drop probability p_{max}, which defines the slope of the drop probability curve between the thresholds b_{min} and b_{max} [7], [8]. It is generally unclear how the parameter should be set, especially when the characteristics of the TCP flow population are unknown. If p_{max} is too large for the current traffic mix, small AQL increases can cause excessive packet drop rates and likely the onset of global synchronization conditions with sustained buffer underflow effects. If p_{max} is too small, the AQL can easily exceed b_{max} and again create the conditions for global synchronization of the TCP sources.

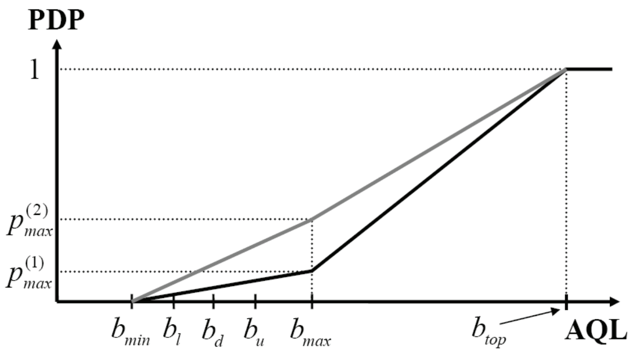

Figure 2. Adjustable packet drop probability curve of Adaptive RED.

The Adaptive RED (ARED) algorithm, specified in [7] as a refinement of a similar concept previously presented in [8], subjects p_{max} to a control algorithm that sets its value within a pre-defined range. To ensure that slow reactions of the control loop to variations in traffic conditions do not trigger packet drop sequences of excessive density, which may induce global synchronization, ARED adopts a gentle version of the drop probability curve, which grows linearly between (b_{max}, p_{max}) and $(b_{top} = 2b_{max}, 1)$ instead of jumping immediately from (b_{max}^{-}, p_{max}) to $(b_{max}^{+}, 1)$ (see Fig. 2). After holding the same p_{max} value for at least a time T ($500\,ms$ is the value of T recommended in [7]), the control algorithm reduces p_{max} as soon as the AQL exceeds a threshold b_u, or increases it as soon as the AQL drops below a threshold b_l, with the ultimate goal of settling the AQL around $b_d = C \cdot d$, where d is the target average delay ($b_{min} < b_l < b_d < b_u < b_{max}$). The authors of [7] automatically derive all buffer thresholds from the target average delay d, relieving the user from the uncertainty of their configuration: $b_{min} = 0.5\,b_d$, $b_l = 0.9\,b_d$, $b_u = 1.1\,b_d$, and $b_{max} = 1.5\,b_d$. The range of allowed p_{max} values is also fixed: $[0.01, 0.5]$. By following all recommendations for default values, the user is left with the target average delay or the desired total allocation of buffer space $Q_{max} \geq b_{top}$ as the only arbitrary parameter. In a nutshell, ARED combines the native

RED with a mechanism for controlling the slope of the two portions of the packet drop probability curve, driven by the ultimate goal of mapping the AQL of the target average delay onto an ideal packet drop probability $p^*(t)$ that continuously matches the phantom drop rate $\varphi(t)$. As noted by the authors, the control algorithm specified in [7] is not optimized for speed of convergence. This is not a problem for the propaedeutical study of this paper, which compares algorithms not by the quality of the control of their operational parameters, but by the link utilization that they enable after all parameters become stable, however long the initial transient can be.

C. *Evaluation of Synchronous ARED*

We use the network simulator 2 (ns2) platform [15] to study the link utilization performance of synchronous ARED (simply called ARED from now on). In all experiments we use the dumbbell network topology shown in Fig. 3, which includes a source aggregation node (SAN), a bottleneck node (BNN), and a sink distribution node (SDN). A number N of TCP Reno sources are attached by respective $1\,Gbps$ links to the SAN. The propagation delay of each of these links sets the RTT of the respective TCP flow. The propagation delay of all other links is negligible. The TCP sinks are attached to the SDN, also by $1\,Gbps$ links. All links between network nodes have $40\,Gbps$ capacity, except the bottleneck link from the BNN to the SDN, whose capacity ranges from $4\,Gbps$ to $36\,Gbps$ in $4\,Gbps$ increments in different experiments. The ARED thresholds for the bottleneck queue are automatically configured to fit into the total buffer size $Q_{max} = 32\,MB$: $b_{min} = 5.3\,MB$, $b_l = 9.54\,MB$, $b_d = 10.6\,MB$, $b_u = 11.66\,MB$, $b_{max} = 15.9\,MB$, and $b_{top} = 31.8\,MB$. Our choice of link rates and buffer thresholds matches the technologically feasible goal of embedding a $32\,MB$ DRAM in the traffic management ASIC of a $40\,Gbps$ line card, as set forth in [3].

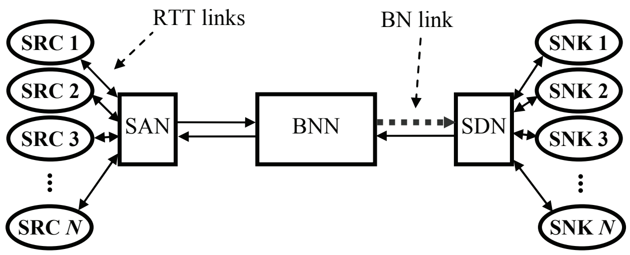

Figure 3. Network configuration for all simulation experiments.

In this section and in the rest of the paper we present results obtained from the simulation of two distinct traffic scenarios, selected out of a much larger set that produces no exceptions to the key outcomes that we are going to illustrate. Scenario 1 has $N = 1000$ TCP flows, all with $\vartheta = 200\,ms$. The number of flows and the RTT value, both relatively large, challenge the ability of the early detection algorithm to distribute the packet losses adequately. We also expect to observe heavy queue length oscillations induced by ordinary TCP dynamics. If packets are dropped too sparsely, the queue will easily overflow. If packets are dropped too frequently, global synchronization will occur. In Scenario 2 we downsize the TCP flow population to $N = 100$ and distribute the RTT values uniformly between $10\,ms$ and $290\,ms$ ($\bar{\vartheta} = 150\,ms$) in order to stress the accuracy of the early detection scheme in the approximation of the phantom drop rate (a single packet drop event has sizable impact on the overall traffic load). The two scenarios are also designed to test the robustness of the buffer management scheme when the key assumptions underlying the small-buffer rule of [3] are individually violated: in Scenario 1 the TCP flows are numerous but synchronized, while in Scenario 2 they are desynchronized but their number is small.

In Fig. 4 we plot a $100\,s$ snapshot of the steady-state evolution of the IQL and AQL at the ARED bottleneck queue in Scenario 1 (bottleneck rate $r = 32\,Gbps$). Note that we consider the queue in steady state when the slow-start thresholds of the TCP sources have started oscillating after dropping from initial oversized values and p_{max} has narrowed the range of its variations. The link utilization measured over the $100\,s$ period is 79.6%. The maximum drop probability p_{max} never moves from the minimum allowed value (0.01) during the entire interval. Fig. 4 explains the loss of link utilization with the periodic onset of global synchronization conditions. We remark that the excessive packet losses that cause the synchronization are always the result of RED decisions and never the consequence of a buffer overflow (the IQL is always far below $Q_{max} = 32\,MB$).

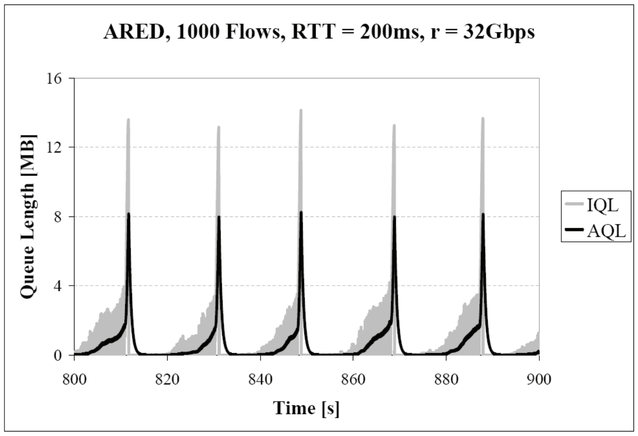

Figure 4. IQL and AQL with default ARED configuration (100*s* interval).

We turn to the finer time granularity of Fig. 5 to find striking evidence that the monotonic nature of the packet drop probability function is the ultimate cause of RED's instability. In Fig. 5 the IQL peaks around time $t_p = 848.77\,s$. By that time, the conditions for a global synchronization event have already been set by excessive packet losses. We know that excessive losses have occurred before t_p because the IQL quickly drops to 0 soon after t_p. This implies that there is an equilibrium time $t_e < t_p$ when the packet drop probability matches the phantom drop rate at a value that could stabilize the queue length if held for an extended period of time. The actual placement of t_e is irrelevant to our argument: all that matters is that the equilibrium drop probability is certainly met before the IQL starts falling. The AQL, which RED uses for driving the packet-drop decisions, systematically trails the IQL at times when the IQL is growing, by a delay that depends on the cutoff frequency of the low-pass filter that computes the AQL (in the plot, the AQL trails the IQL by less than $500\,ms$). The AQL keeps increasing as long as it is smaller than the IQL, therefore also after the equilibrium time t_e (and even after t_p). As the AQL keeps growing, the packet drop probability also grows by effect of the monotonic increasing profile of the drop probability function. This way the packet drop probability remains above the equilibrium value for a time that extends beyond t_p. The packet losses that occur between t_e and t_p, in excess of the losses required for the equilibrium by the phantom drop rate, are those responsible for the onset of the global synchronization event. A plot of the samples collected for the time distance between consecutive packet drop events during the same interval of Fig. 5, not shown here for space reasons, confirms that RED keeps increasing the packet drop frequency way beyond t_p, causing the onset of global synchronization.

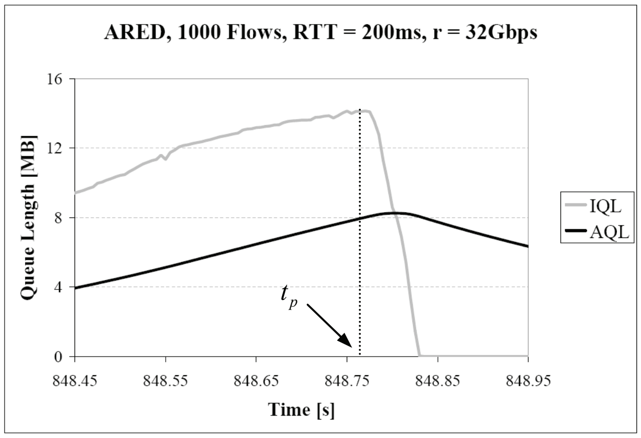

Figure 5. IQL and AQL with default ARED configuration ($0.5\,s$ interval).

The behavior shown in Figs. 4 and 5 is caused by the inability of RED to lock on the equilibrium condition after reaching it, which is a product of the strictly increasing nature of the drop probability function. Our experiments with Scenario 2 provide identical evidence. However, before we can generalize the conclusion that the monotonic increasing profile of the drop probability function is the ultimate cause of global synchronization and of the subsequent loss of link utilization under most traffic conditions, we must verify that our results are not just the product of a bad choice of ARED parameters, and specifically of a minimum allowed value of p_{max} ($p_{max}^{(l)}$) that is too large for the specific traffic mixes of our experiments ($p_{max}^{(l)} = 10^{-2}$ in the default ARED configuration).

To verify the critical role of $p_{max}^{(l)}$, we run the two scenarios again with $p_{max}^{(l)} = 10^{-4}$ and $p_{max}^{(l)} = 10^{-6}$. In Fig. 6 we plot the steady-state link utilization measurements for both scenarios with $p_{max}^{(l)} \in \{10^{-2}, 10^{-4}, 10^{-6}\}$. The plots show that the ARED performance improves with $p_{max}^{(l)} = 10^{-4}$, but then degrades again with $p_{max}^{(l)} = 10^{-6}$. In experiments that yield link utilization below 99% we observe that p_{max} tends to settle on the $p_{max}^{(l)}$ value. With $p_{max}^{(l)} = 10^{-4}$ all packet drop events are triggered while the AQL is below b_{max}, just like in the $p_{max}^{(l)} = 10^{-2}$ case, but the queue length peaks are much more frequent than in Fig. 4, because the excess losses that occur after t_e are in lower numbers and their long-term global synchronization effect is almost null. This is a sign that the ideal packet drop probability is somewhere below 10^{-4} but not far from it. Instead, with $p_{max}^{(l)} = 10^{-6}$ most losses occur when the AQL is above b_{max}, a clear symptom that p_{max} is too small (and $p_{max}^{(l)}$ with it). We can deduct that the control algorithm that sets p_{max} in ARED is flawed, because it fails to keep p_{max} above the ideal value required by the phantom drop rate. This happens again because of the monotonic increasing nature of RED's control law. When p_{max} is lower than needed, the AQL enters the "gentle" region between b_{max} and b_{top}, where global synchronization is hardest to avoid (the slope of the packet drop probability is typically much higher than in the region between b_{min} and b_{max}). Hence there is no time for p_{max} to accumulate any relevant increase before being pushed down again as global synchronization empties the queue once more.

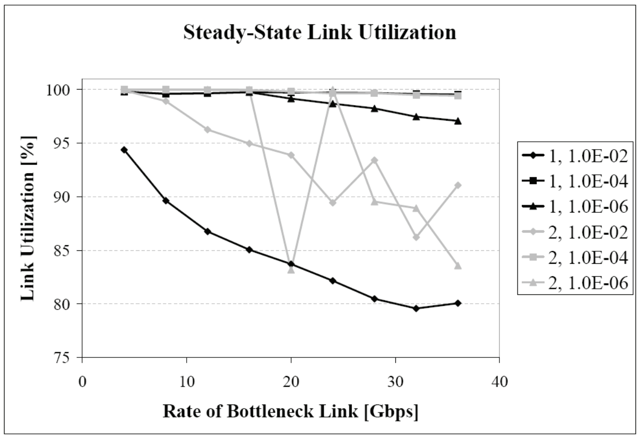

Figure 6. Steady-state link utilization with ARED; the plot labels identify the simulation scenario (1 or 2) and the value of $p_{max}^{(l)}$.

From our examination of RED we extract three observations that drive the design of our new early detection scheme. First, it is possible to improve the performance of RED by proper adjustment of the p_{max} value. However, ARED is unfit for the task, whether or not we stick to the original guidelines [7] for configuration of $p_{max}^{(l)}$. Second, the performance of RED improves more evidently when the ideal value of p_{max} is extremely low (somewhere below 10^{-4} in Scenarios 1 and 2). Third, the experiments with $p_{max}^{(l)} = 10^{-6}$ prove that packet drop probability values far lower than the ideal one not only serve no purpose, but are actually detrimental, because they delay the queue response to the onset of congestion conditions. From the second and third observations we deduct that it is not just the value of p_{max} that determines the chances of success of a RED configuration, but also the slope of the drop probability curve, which should be extremely low and possibly even null.

III. THE PERIODIC EARLY DETECTION ALGORITHM

In this section we present the algorithmic core of a new early detection scheme that builds on the idea that a stable instance of RED is one that associates a *fixed* packet drop probability with an extended range of AQL values, so that ordinary increases of the AQL (as it catches up with the IQL) do not induce higher packet drop rates and global synchronization. To consistently enforce the desired packet drop rate, we replace the notion of packet drop probability, which yields variable inter-drop intervals, with a packet drop period that enforces equally spaced packet drop events. For this reason we refer to our scheme as Periodic Early Detection (PED). PED combines two components that operate at different timescales. At the shorter (packet) timescale, PED drops packets at fixed time intervals when signs of congestion are evident. At the longer (RTT) timescale, a control algorithm adjusts the packet drop period to the evolution of the AQL.

PED uses a drop timer with period τ_D of controllable duration to trigger the sampling of the IQL q and AQL $\overline{q}$ and their comparison with respective thresholds b_{min}^{PED} and b_{gate}^{PED} ($b_{min}^{PED} > b_{gate}^{PED}$). If $q > b_{min}^{PED}$ AND $\overline{q} > b_{gate}^{PED}$ when the drop timer expires, PED drops the next incoming packet; otherwise it accepts into the queue the next packet and all the packets that follow, up to the next expiration of the drop period. (For simplicity of presentation, we describe here the packet version of the early detection algorithm; just like for RED, the definition of a byte version of PED is straightforward.)

PED controls the period τ_D of the drop timer based on the AQL evolution. At time intervals never shorter than a time constant T that is large enough to include the RTT values of most TCP connections (e.g., $T = 500\,ms$), PED compares $\overline{q}$ with the minimum PED threshold b_{min}^{PED} and a maximum PED threshold b_{max}^{PED}. PED increases τ_D if $\overline{q} < b_{min}^{PED}$ and decreases it if $\overline{q} > b_{max}^{PED}$. In both cases, the new value of the drop period $\tau_D[m]$ at correction event m is derived from both the current value $\tau_D[m-1]$ (set at correction event $m-1$) and the previous value $\tau_D[m-2]$ (set at correction event $m-2$). The size of the period correction is modulated by the ratio between the AQL and the relevant threshold ($\overline{q} / b_{min}^{PED}$ for period increases and $b_{max}^{PED} / \overline{q}$ for period decreases). The period of the drop timer remains unchanged every time the AQL is found to be in between the two thresholds.

The pseudo-code of Fig. 7 summarizes the update of the packet drop period after at least a time T has elapsed since the latest drop period update. In the equations of Fig. 7, K is the maximum size of the period correction (the correction is maximum when $\alpha = 0$).

```
if $\left(\overline{q} < b_{min}^{PED}\right)$ then
    $\alpha = \overline{q} / b_{min}^{PED}$
    $\tau_D[m] = \min\{\tau_{D,max}, \alpha\, \tau_D[m-1] + (1-\alpha)\cdot K \cdot \tau_D[m-2]\}$
else if $\left(\overline{q} > b_{max}^{PED}\right)$ then
    $\alpha = b_{max}^{PED} / \overline{q}$
    $\tau_D[m] = \max\{\tau_{D,min}, \alpha\, \tau_D[m-1] + (1-\alpha)\cdot \tau_D[m-2] / K\}$
```

Figure 7. Pseudo-code for drop period update in PED.

PED uses a synchronous, time-driven background process for updating the AQL. The criteria for setting the averaging period τ_q are the same that hold for the synchronous versions of RED: the period should be larger than the inter-departure time of packets of typical size at the full capacity of the link (e.g., $\tau_q \geq 1500\,B / 40\,Gbps = 0.3\,\mu s$), but not larger than a small fraction (e.g., 5%) of the target average delay (e.g., $\tau_q \leq 0.05 \cdot 1\,ms = 50\,\mu s$). As usual, PED computes the AQL $\overline{q}$ as an EWMA: $\overline{q}[n] = w \cdot q[n] + (1-w) \cdot \overline{q}[n-1]$. The EWMA weight w is defined by the ratio between the averaging period and the time constant of the TCP data path: $w = \tau_q / T$.

In order to prevent ordinary TCP dynamics from diverting the control of the drop period from its goal of matching the phantom drop rate, PED also includes provisions for: (a) suspending the corrections of the drop period τ_D under low-load conditions that are not the consequence of recent packet drop events; (b) resetting the drop period to the minimum value available after the buffer occupancy grows from empty to full within a time that is comparable with the time constant T (such an event is a sign that the current packet drop period is too large for dealing properly with the current traffic mix); and (c) allowing emergency corrections of the drop period even before expiration of the time constant T as soon as the AQL exceeds a safety threshold $b_{safe}^{PED} > b_{max}^{PED}$.

PED differs from RED in many ways. As already discussed, PED keeps the packet drop rate fixed for a minimum time T instead of changing it continuously with the AQL. It minimizes the variations in the inter-drop times by replacing the packet drop probability with a fixed packet drop period. The synchronization of the averaging process is also not included in the canonical versions of RED and ARED [6], [7], but others have already proposed it in the past [1], [10].

Another important element of novelty in PED is the careful consideration of the effects that data path and filtering delays have on the interaction between the buffer management scheme and the TCP source dynamics. It takes a time comparable with the RTT for a source to recognize a packet loss and for that recognition to produce visible effects on the IQL of the bottleneck queue. Because of the EWMA with time constant $T \geq \overline{\vartheta}$, it takes a similar extra time for the AQL to catch up with the IQL variation. The accuracy of the control mechanism that tracks the phantom drop rate depends tightly on the time distance between the adjustments in the activity of the TCP sources and the corrective actions on the queue length that drive those adjustments. While the delay induced by the RTT cannot be avoided, PED excludes the extra delay contribution of the EWMA by giving the IQL the prominent role in defining the packet drop decision. It is true that PED also checks the AQL to confirm that early signs of congestion are present, but the threshold b_{gate}^{PED} used for this purpose is only half the size of the main threshold b_{min}^{PED}, so that the EWMA delay has practically no impact on the decision. Similarly, in setting the packet drop period at correction event m we let $\tau_D[m-2]$ give an extra contribution to $\tau_D[m]$ (besides the one already included in $\tau_D[m-1]$) because the state of the queue that is observed at event m (i.e., $(q(t_m), \overline{q}(t_m))$) may depend on $\tau_D[m-2]$ much more than on $\tau_D[m-1]$. In fact, the sources affected by the packet losses triggered by drop period $\tau_D[m-1]$ may not even have started reacting to those losses by the time $\tau_D[m]$ is set.

IV. PED CONFIGURATION

The following is the list of all the configuration parameters that drive the operation of PED, inclusive of setting recommendations that we obtained empirically and then found consistently validated in all of our experiments: (a) Q_{max} is the total buffer space available; its value is set by hardware design constraints, such as the size of the available buffer memory (e.g., $Q_{max} = 32\ MB$ for the on-chip implementation of buffer memories in our Scenarios 1 and 2); (b) b_{min}^{PED} is the minimum PED threshold; we configure it as a fixed fraction (20%) of Q_{max} (e.g., $b_{min}^{PED} = 6.4\ MB$ in our simulation scenarios); (c) b_{max}^{PED} is the maximum PED threshold; it should be twice as large as the minimum PED threshold, but other values higher than b_{min}^{PED} are accepted (e.g., $b_{max}^{PED} = 12.8\,MB$); (d) b_{safe}^{PED} is the safety PED threshold; it should be three times as large as the minimum PED threshold, but other values higher than the maximum PED threshold are accepted (e.g., $b_{safe}^{PED} = 19.2\ MB$); (e) b_{gate}^{PED} is the gating PED threshold; no packet is dropped by PED as long as the AQL is below this threshold; it should be half the size of b_{min}^{PED} (e.g., $b_{gate}^{PED} = 3.2\ MB$); (f) τ_q is the update period for the AQL; it should be large enough to avoid multiple updates of $\overline{q}$ while the same packet is in transmission out of the queue (e.g., $\tau_q = 10\ \mu s$); (g) T is the time constant of the control system made of the bottleneck link and the set of TCP sources whose packets traverse the link; it is also the inverse of the cutoff frequency of the low-pass filter that implements the computation of $\overline{q}$; to make sure that the RTT values of most TCP connections are included, especially when their actual distribution is unknown, the value of T should be set always to $500\ ms$; lower values are accepted when the RTT distribution is known to be concentrated around a definitely smaller value; (h) w is the weight used in the EWMA computation of the AQL; its value derives directly from the averaging period τ_q and the time constant T: $w = \tau_q / T$ (e.g., $w = 0.00002$); (i) $\tau_D^{(l)}$ is the minimum admitted PED drop period; it should never be less than the averaging period τ_q (e.g., $\tau_D^{(l)} = 100\ \mu s$); (j) $\tau_D^{(u)}$ is the maximum admitted PED drop period; it should be larger than $\overline{\vartheta}$, but not larger than T (e.g., $\tau_D^{(u)} = 500\,ms$); and (k) K is the fixed correction factor used for updating the packet drop period τ_D (e.g., $K = 2$; larger values make the control loop faster but less stable).

Since we offer fixed recommendations for the values of all parameters, we can claim that the configuration of PED is straightforward and can be fully automated once the link capacity and the amount of available memory are known.

V. PED PERFORMANCE

In the simulation of Scenarios 1 and 2 we configure the PED parameters with the values assigned in the examples of Section IV: $\tau_q = 10\,\mu s$, $T = 500\,ms$, $\tau_D^{(l)} = 100\ \mu s$, $\tau_D^{(u)} = 500\ ms$, $Q_{max} = 32\,MB$, $b_{gate}^{PED} = 3.2\,MB$, $b_{min}^{PED} = 6.4\,MB$, $b_{max}^{PED} = 12.8\,MB$, and $b_{safe}^{PED} = 19.2\ MB$.

With Scenario 1 the measured steady-state link utilization is 100% for all values of bottleneck rate. Fig. 8 shows the evolution of the IQL and AQL over a $100\,s$ interval with bottleneck rate $r = 32\,Gbps$. The plots emphasize the critical role of a stable packet drop period in preserving the long-term stability of the queue (at steady state the PED drop period oscillates narrowly around $20\,ms$).

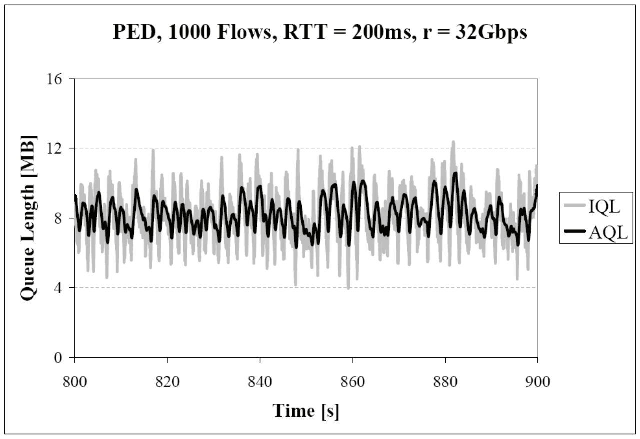

Figure 8. IQL and AQL with PED (Scenario 1, $r = 32\,Gbps$, 100s interval).

Under Scenario 2 we measure the minimum steady-state utilization of the bottleneck link (99.947%) when the bottleneck rate is $36\,Gbps$. Fig. 9 plots the IQL and AQL over a $10\,s$ interval. Compared to Scenario 1, the width and frequency of the IQL oscillations increase substantially, but PED still manages to keep the link utilization close to 100%. It is important to remark that at steady state the PED drop period settles almost permanently (there are only sporadic, short-lived exceptions) on the maximum allowed value of $500\,ms$, indicating that the PED control loop would likely push the value higher if a wider range was available. However, the value limitation on the maximum PED drop period does not compromise the link utilization performance, because a proper time separation between subsequent packet drop events is still enforced by the gating of the packet drop decision through the comparison of the AQL with the gating threshold b_{gate}^{PED}.

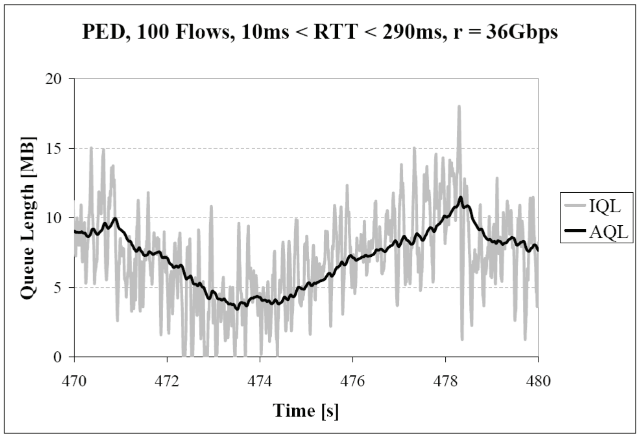

Figure 9. IQL and AQL with PED (Scenario 2, $r = 36\,Gbps$, 10s interval).

When we scale the full capacity of the bottleneck link down ($10\,Gbps$) and up ($100\,Gbps$) and accordingly modulate the bottleneck rate (from $1\,Gbps$ to $9\,Gbps$ in $1\,Gbps$ increments in the former case, from $10\,Gbps$ to $90\,Gbps$ in $10\,Gbps$ increments in the latter), we observe that the link utilization performance of PED remains unchanged if we also scale linearly all the buffer thresholds ($Q_{max} = 8\,MB$ and $Q_{max} = 80\,MB$, respectively). This linear dependency retains the 2.5% buffer-space ratio between PED and the BDP rule that we have applied throughout this section to the $40\,Gbps$ case.

VI. Conclusions

With ever-increasing transmission rates in network links, the on-chip implementation of packet buffers is a primary requisite for the scalability and energy efficiency of routers and switches. Existing buffer management approaches such as Tail Drop and RED do not enable the necessary reductions of buffer space because they fail to avoid the global synchronization of TCP sources under common traffic scenarios. RED also suffers from the lack of a configuration strategy that guarantees high-end performance irrespective of the traffic mix.

We have shown that the main reason for RED's shortcomings is the monotonic non-decreasing profile of the control law that derives the frequency of the packet drop events from the queue length. Accordingly, we have defined a new Periodic Early Detection (PED) scheme where the control law is flat, at a level that is adjusted at the RTT timescale. We have collected simulation results that assert PED's capability to consistently enforce 100% link utilization with long-lived TCP flows with only 2.5% of the memory space used in current designs, in scenarios with average RTT up to 500 *ms*.

In practical applications, the traffic mix in a bottleneck link is not made only of long-lived TCP flows, but also includes UDP and short-lived TCP flows (i.e., TCP flows that have not left the slow-start phase [3]). Those flows respond to packet losses differently than long-lived TCP flows. Simple experiments with PED reveal that the throughput of TCP flows and the utilization of the bottleneck link degrade heavily if TCP and UDP packets are all stored in one queue. However, this is true of any buffer management scheme and is not specifically a PED issue. Moreover, the enforcement of bandwidth and delay guarantees for applications that rely on UDP transport already mandate the assignment of TCP and UDP packets to different queues. Differentiation in queue treatment between short-lived and long-lived TCP flows has been advocated in other contexts [16] and may also prove beneficial in PED queues.

Topics for future work include the exploration of traffic management options for the coexistence of different traffic types within the boundaries of on-chip buffer implementations, and the optimization of the control component of the PED algorithm for accuracy and speed of convergence.

References

[1] V. Jacobson, K. Nichols, and K. Poduri, "RED in a different light," <http://www.cnaf.infn.it/~ferrari/papers/ispn/red_light_9_30.pdf>, 1999.

[2] C. Villamizar and C. Song, "High-performance TCP in ANSNET," ACM SIGCOMM Comp. Communications Review, 24(5):45-60, 1994.

[3] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," Proceedings of ACM SIGCOMM 2004.

[4] A. Vishwanath, V. Sivaraman, and M. Thottan, "Perspectives on router buffer sizing: Recent results and open problems," ACM SIGCOMM Computer Communication Review 39(2):34-39, 2009.

[5] Y. Ganjali and N. McKeown, "Update on buffer sizing in Internet routers," ACM SIGCOMM Computer Communication Review, 36(5):67-70, 2006.

[6] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, 1(4):397-413, 1993.

[7] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management," <http://icir.org/floyd/papers/adaptiveRed.pdf>, 2001.

[8] W. Feng, D. Kandlur, D. Saha, and K. Shin, "A self-configuring RED gateway," Proceedings of IEEE Infocom 1999.

[9] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED," Proceedings of IEEE INFOCOM 1999.

[10] V. Misra, W. B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," Proceedings of ACM SIGCOMM 2000.

[11] V. Jacobson, "Notes on using RED for queue management and congestion avoidance," Nanog 13 Workshop, Dearborn, MI, June 1998.

[12] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," IETF RFC 3168, September 2001.

[13] M. May, J. Bolot, C. Diot, and B. Lyles, "Reasons not to deploy RED," Proceedings of IEEE/IFIP IWQoS 1999, London, UK, June 1999.

[14] R. Braden et al., "Recommendations on queue management and congestion avoidance in the Internet," IETF RFC 2309, April 1998.

[15] ns2. <http://nsnam.isi.edu/nsnam/index.php/Main_Page>.

[16] D.M. Divakaran, G. Carofiglio, E. Altman, and P. Vicat-Blanc Primet, "A flow scheduler architecture," Lecture Notes in Computer Science, Vol. 6091, Networking 2010, 122-134, SpringerLink, 2010.

Energy-aware Scheduling Algorithms for Network Stability

Matthew Andrews
Bell Labs, Murray Hill, NJ
andrews@research.bell-labs.com

Spyridon Antonakopoulos
Bell Labs, Murray Hill, NJ
spyros@research.bell-labs.com

Lisa Zhang
Bell Labs, Murray Hill, NJ
ylz@research.bell-labs.com

***Abstract*—A key problem in the control of packet-switched data networks is to schedule the data so that the queue sizes remain bounded over time. Scheduling algorithms have been developed in a number of different models that ensure network stability as long as no queue is inherently overloaded. However, this literature typically assumes that each server runs at a fixed maximum speed. Although this is optimal for clearing queue backlogs as fast as possible, it may be suboptimal in terms of energy consumption. Indeed, a lightly loaded server could operate at a lower rate, at least temporarily, to save energy.**

Within an energy-aware framework, a natural question arises: "What is the minimum energy that is required to keep the network stable?" In this paper, we demonstrate the following results towards answering that question.

Starting with the simplest case of a single server in isolation, we consider three types of rate adaptation policies: a heuristic policy, which sets server speed depending on queue size only, and two more complex ones that exhibit a tradeoff between queue size and energy usage. We also present a lower bound on the best such tradeoff that can possibly be achieved.

Next, we study a general network environment and investigate two scenarios. In a temporary sessions scenario, where connection paths can rapidly change over time, we propose a combination of the above rate adaptation policies with the standard Farthest-to-Go scheduling algorithm. This approach provides stability in the network setting, while using an amount of energy that is within a bounded factor of the optimum. In a permanent sessions scenario, where connection paths are fixed, we examine an analogue of the well-known Weighted Fair Queueing scheduling policy and show how delay bounds are affected under rate adaptation.

I. Introduction

In this paper we revisit a number of classical network scheduling problems and examine how they are affected when we introduce energy awareness. Most such scheduling problems have been formulated under the assumption that the processing rate of a server is fixed. However, modern servers often allow for dynamic adjustment of their processing rate, because operating at a lower rate typically requires less power. This has inspired a growing body of research on how to best take advantage of this so-called *rate adaptation* capability in the interest of maximizing energy savings.

Herein, our main focus is whether it is possible to maintain stability, or equivalently bounded queues, in a network of servers in an energy-aware manner. In particular, we investigate the tradeoff between energy consumption and performance measures such as delay and queue size.

We remark that there exists an extensive literature proposing various scheduling algorithms that keep the network stable, as long as the servers are always operating at maximum rate. In order to study the problem we posed earlier, though, our task is to determine how to adapt these algorithms so as to minimize energy while preserving stability. The related objective of guaranteeing bounded end-to-end packet delay is also considered.

A. Modeling

First of all, let us describe our modeling of the problem. We consider a network of multiple servers, which process and then forward incoming data traffic. Every server's processing rate may be adjusted independently, taking any value in the interval $[R_{\min}, R_{\max}]$[1], where $R_{\min}$ and $R_{\max}$ are given parameters such that $0 < R_{\min} < R_{\max}$. Furthermore, the power consumed by the server e while operating at rate r_e is given by a so-called *energy function* $f(r_e)$. We adopt the common assumption that $f(s) = s^\alpha$ for some parameter $\alpha > 1$, which is based on properties of CMOS circuits [9], [16].

For every server, a *rate-adaptive scheduling algorithm* makes two decisions at any given time, namely setting the processing rate and determining which data in the server queue to forward. We concentrate on *work-conserving* algorithms only; in other words, the algorithm cannot order a server not to forward any traffic (for whatever reason) at a time when the latter's queue is non-empty. Moreover, with regard to network traffic we consider two standard models, defined below.

1) Temporary Sessions Model: This model is also known as the Adversarial Queuing Model (AQM). It aims to capture situations in which the set of sessions (or connections) carried by the network is highly volatile. The model stipulates that packets are injected into the network by some adversarial process $\mathcal{A}$, henceforth referred to simply as the adversary. Additionally, $\mathcal{A}$ specifies the path along which each packet must be routed at the time of its injection, and the hop count of every such path is bounded by a parameter d. Let $A_e(t, t')$ be the total size of packets injected into the network during the time interval $[t, t')$ that include server e on their paths. In order for each server not to be inherently overloaded, the packet injection by $\mathcal{A}$ is restricted to be $(\sigma, 1-\varepsilon)$-*admissible*, which means that for all e and all intervals $[t, t')$,

$$A_e(t, t') \leq \sigma + (1-\varepsilon) R_{\max}(t' - t) \ ,$$

[1]In practice, only a discrete set of rates would likely be available, due to hardware constraints. Nevertheless, we extend it to a continuous range for clarity of exposition. This entails no loss of generality for our results.

where $\sigma \geq 0$ is the burst size and $\varepsilon > 0$ reflects the load factor. We then say that $\mathcal{A}$ is a *bounded adversary of rate* $(\sigma, 1-\varepsilon)$.

2) Permanent Sessions Model: This model is sometimes referred to as the connection-oriented model, and it reflects a setting where all data is transported along pre-defined connections. Each connection i is specified by a path P_i, a burst size σ_i, and an injection rate ρ_i. Let $H_i(t,t')$ be the total injection into connection i during the time interval $[t,t')$. Again, to ensure servers are not inherently overloaded, the injection is restricted to be admissible in the following sense.

$$\begin{aligned} H_i(t,t') &\leq \sigma_i + \rho_i(t'-t) && \forall i, \forall [t,t') \\ \sum_{i:e\in P_i} \rho_i &\leq (1-\varepsilon)R_{\max} && \forall e \end{aligned} \tag{1}$$

Remark. It is easy to see that any traffic admissible to the permanent sessions model is also admissible to the temporary sessions model. However, the converse statement is not true. Indeed, consider two paths P_1 and P_2 that share a server e. Let us partition time into intervals of arbitrarily large lengths. The temporary sessions model allows injections that alternate between only injecting along P_1 during odd-numbered intervals and only injecting along P_2 during even-numbered intervals, at rate $0.9R_{\max}$ in both cases. This cannot happen in the permanent sessions model, because the connection rates must be set at $\rho_1 = \rho_2 = 0.9R_{\max}$. As a result, $\sum_{i:e\in P_i} \rho_i > R_{\max}$, which would violate (1). (Note that since the interval lengths are arbitrarily large, admissibility in the permanent sessions model cannot be achieved by setting a large burst size for each connection.) Therefore, strictly more traffic injections are admissible in the temporary sessions model.

We say that the network is *stable* if the aggregate queue size remains bounded over time. Our goal in this paper is to maintain stability while also tailoring server rates to traffic loads in such a way that energy usage is minimized. For example, if the long-term traffic through server e satisfies $A_e(t,t') \ll R_{\max}(t'-t)$ for $t' \gg t$, then the latter can operate at a rate much smaller than $R_{\max}$ without jeopardizing stability. However, there is no way to know exactly how much traffic will arrive at e in the future, and thus it is not clear *a priori* how to set its rate for optimal energy efficiency. This constitutes a scheduling problem which, to the best of our knowledge, has not been addressed in conjunction with network stability before.

B. Previous Work

1) Network stability: If every server runs at rate $R_{\max}$ constantly, then there exist well-known scheduling algorithms ensuring a time-independent upper bound on the size of all queues, in the temporary sessions model, for any given bounded adversary and any network topology. Such algorithms are called *universally stable*. More specifically, it was shown in [1] that several scheduling algorithms – including Farthest-to-Go (FTG) and Nearest-to-Source (NTS), whose definitions we provide later – are universally stable, and also guarantee bounded end-to-end packet delay. By contrast, some very natural algorithms such as First-in-First-out (FIFO) and Last-in-First-out (LIFO) are not universally stable.

In the permanent sessions model, the most widely studied scheduling algorithm is Generalized Processor Sharing (GPS), in particular its packetized form that is sometimes called Weighted Fair Queueing (WFQ) [8]. Under GPS and WFQ, each server operates by splitting service among all backlogged connections according to some predetermined weights. We shall focus on the version known as Rate Proportional Processor Sharing (RPPS), in which the weight for connection i is equal to ρ_i at each server. Parekh and Gallager [13], [14] proved that if each server always runs at rate $R_{\max}$ then the packetized version of RPPS is stable and they derived an end-to-end delay bound for each connection. (See (2) in Section I-C3.) We note that RPPS highlights the difference between the two traffic models in question, since it has been established that RPPS is *not* necessarily stable in the temporary sessions model [3].

2) Energy Efficiency: The study of energy minimization via rate adaptation was initiated by Yao et al. [16], where they considered energy functions of the form $f(x) = x^\alpha$. Subsequently, a large number of papers focused on the problem of conserving energy on a single server. Most of this prior work falls in two categories. In the first one (e.g. [5], [7]), every job has an associated deadline, and the goal is to minimize energy while meeting all the deadlines. In the second category (e.g. [4], [15]), jobs do not have individual deadlines and the goal is to minimize the sum of the energy used plus the aggregate response time of the jobs. Note that even in the single-server case neither of these objectives directly addresses our goal of minimizing energy consumption while maintaining stability.

Another body of work focuses on the *powerdown* model, in which the servers cannot alter their processing rates but can toggle between the on and off states at a switching cost. For example, [10] discusses the energy consequences of putting router and switch components to sleep. The survey [11] presents known results for both the powerdown and the rate adaptation models.

As already mentioned, much less attention was paid to scheduling for energy minimization in *networks* of servers. Nedevschi et al. [12] considered both rate adaptation and powerdown in the context of multiple servers and concluded that energy could be saved if we batch together packets with the same source and destination. Alternative techniques for batching packets in the powerdown model and thereby minimizing the number of transmissions between states were also explored in [2]. To the best of our knowledge, no prior work has attempted to tackle the specific problem that we deal with here, namely that of minimizing energy usage while maintaining network stability.

C. Results

Our paper is divided into three main sections.

1) Single-server results: In Section II we focus on scheduling a single server in isolation. This allows us to study basic issues such as the tradeoff between queue sizes and energy in this simplest setting. For example, suppose $\mathsf{opt}_B(t)$ is the optimal energy necessary to keep the queue size bounded by B up to time t. We show that no online algorithm can simultaneously keep the energy consumption within a certain factor of opt_B *and* the queue size within a corresponding constant factor of B. This establishes a concrete limit on what one can hope for in terms of maintaining stability while minimizing energy.

We then propose three rate-adaptive policies to set server rates: Batch, SlowStart and queue-based. Roughly speaking, the Batch policy accumulates data of size $2B$ before serving it at a rate equal to the average arrival rate of this data. This policy ensures a maximum queue size of $O\big(B\log_2\frac{R_{\max}}{R_{\min}}\big)$ and energy usage of $O(6^\alpha\cdot\mathsf{opt}_B(t))$. The SlowStart policy starts with rate $R_{\min}$ at the beginning of each interval where the queue is non-empty, and after some time it ramps up the processing rate linearly, in a deterministic fashion. SlowStart ensures a maximum queue size of $O\big(\sigma+B\frac{R^2_{\max}}{R^2_{\min}}\big)$ and energy consumption of $O(2^\alpha\cdot\mathsf{opt}_B(t))$. The queue-based policy (or rather family of policies) sets the server rate solely as a function of the current queue size. Although we cannot bound the energy usage of this approach for all admissible traffic, when the arrival rate is constant it can keep the queue size bounded by $O(B)$ while consuming $O(\mathsf{opt}_B(t))$ energy.

2) Results for Temporary Sessions Model: In Section IV we examine the multiple-server scenario in the temporary sessions model. Our starting point is the universally stable [1] algorithm Farthest-to-Go (FTG), which gives priority to data farthest from its destination (in terms of server hops). We generalize all three rate-adaptive policies mentioned above to multiple servers. When combined with FTG, both Batch and SlowStart yield the desired properties of bounded queue size and bounded energy consumption. Additionally, we derive a bound on end-to-end delay in the case of SlowStart, although we are so far unable to establish a similar claim for Batch.

Note that for "traditional" work-conserving scheduling algorithms, bounded delay is equivalent to bounded queue (hence stability). However, this is no longer true when rate adaptation is an option, as demonstrated by the example in Section III.

In the interest of space, our presentation focuses on combining SlowStart with FTG. All results in this section carry over if we replace FTG with another universally stable algorithm Nearest-to-Source (NTS), which gives priority to data closest to its source.

3) Results for Permanent Sessions Model: For the Permanent Sessions Model, our starting point is the work of Parekh and Gallager [13], [14], who showed that if every server always runs at $R_{\max}$, Weighted Fair Queueing guarantees the following end-to-end delay bound for each connection i,

$$\frac{\sigma_i+(K_i-1)L_i}{\rho_i}+K_i\frac{\max_i L_i}{R_{\max}} \qquad (2)$$

where σ_i, ρ_i, K_i and L_i are respectively the burst size, connection rate, hop count and maximum packet size for connection i. Our major result in this section is that for any rate-adaptive policy that uses rates between $R_{\min}$ and $R_{\max}$ and always uses rate $R_{\max}$ when the queue exceeds a threshold U, if we schedule according to Weighted Fair Queueing then the end-to-end delay is bounded by

$$\frac{\sigma_i+(K_i-1)L_i}{\rho_i}+K_i\left(\frac{\max_i L_i}{R_{\min}}+\frac{R_{\max}}{R_{\min}}\frac{U}{\rho_i}\right).$$

II. The Single-server Case

We begin our analysis by focusing on a single server in isolation. This will allow us to determine some of the basic tradeoffs between queue size and energy usage. First of all, we define a simple lower bound on energy consumption to keep the queue size bounded by B, which shall be used as a benchmark henceforth. We then present a bound on the optimal tradeoff between queue size and energy efficiency that can be achieved by any rate adaptation policy. Subsequently, in Sections II-C, II-D, and II-E, we propose three approaches to set the server rate adaptively and show upper bounds on their energy usage and queue size.

A. Lower bound on energy usage

Recall that $\mathsf{opt}_B(t)$ is the minimum amount of energy required by time t if we wish to keep the queue bounded by B at all times. Let $\mathsf{opt}_B(t,t')$ be defined similarly on the interval $[t,t')$. We derive a simple bound on $\mathsf{opt}_B(t,t')$.

Lemma 1. *For a single server e,*

$$\mathsf{opt}_B(t,t')\ge f\left(\max\left\{R_{\min},\frac{A_e(t,t')-B}{t'-t}\right\}\right)\cdot(t'-t)\,.$$

Proof: Since data of size $A_e(t,t')$ arrives during the time interval $[t,t')$ and the queue size is to be less than B at time t', then the amount of data that must be processed during $[t,t')$ is at least $\max\{0,A_e(t,t')-B\}$. However, we are assuming that $f(\cdot)$ is a convex function and the minimum processing rate is $R_{\min}$, thus in order to serve that amount of data with minimal energy usage the server must operate at speed $\max\{R_{\min},(A_e(t,t')-B)/(t'-t)\}$ throughout the interval $[t,t')$. The bound follows. ■

We shall assume that $B\ge\sigma$, since one cannot hope to maintain the queue size smaller than the burst size. Even then, the energy bound of Lemma 1 may not be achievable, for instance if most of the data injected during $[t,t')$ is injected towards the end of the interval.

B. Bound on tradeoff between queue size and energy

Intuitively, guaranteeing a better bound on queue size should incur higher energy usage. We establish that such a tradeoff is inherently unavoidable.

Lemma 2. *Let* $x_1=\frac{B}{R_{\min}},x_2,x_3,\ldots$ *be a sequence that satisfies*

$$x_j f\left(\frac{B}{2x_j}\right)\ge\nu\sum_{\ell<j}x_\ell f\left(\frac{B}{x_\ell}\right), \qquad (3)$$

for some given ν. Further, suppose that a rate adaptation policy RA *uses energy at most $\nu\cdot\mathsf{opt}_B(t)$ by time t, for all times t. Then, the maximum queue size under* RA *is at least $(J(\nu)+1)B/2$, where $J(\nu)=\operatorname{argmax}\left\{j:x_j\ge\frac{B}{R_{\max}}\right\}$.*

Proof: We define a sequence $t_0 = 0, t_1, \ldots, t_{J(\nu)}$, with $t_j = t_{j-1} + x_j$ for $1 \leq j \leq J(\nu)$. Assume that data of size B arrives at each time $t_0, t_1, \ldots, t_{J(\nu)}$. An optimal rate adaptation policy would serve data of size B during each interval $[t_{j-1}, t_j)$ for $1 \leq j \leq J(\nu)$, so that the queue size remains at most B.

Nevertheless, the policy RA cannot follow this behavior, for the simple reason that it cannot predict the future. More precisely, when data of size B arrives at time t_j, $j \geq 1$, RA can deduce what the optimal schedule should have been for serving all data injected at $t_0, \ldots, t_{j-1}$. Then, by Lemma 1,

$$\mathsf{opt}_B(t_j) \geq \sum_{\ell \leq j} (t_\ell - t_{\ell-1}) f\left(\frac{B}{t_\ell - t_{\ell-1}}\right) = \sum_{\ell \leq j} x_\ell f\left(\frac{B}{x_\ell}\right) .$$

Since RA does not know whether (or when) additional data will be injected in the future, it can only afford to use at most $\nu \cdot \mathsf{opt}_B(t_j)$ energy in the interval $[0, t_{j+1})$, hence *a fortiori* in the interval $[t_j, t_{j+1})$. From (3),

$$\nu \cdot \mathsf{opt}_B(t_j) \leq (t_{j+1} - t_j) f\left(\frac{B/2}{t_{j+1} - t_j}\right) ,$$

which implies that the amount of data RA can serve in $[t_j, t_{j+1})$ is no more than $B/2$. Finally, during the interval $[t_0, t_1)$, RA sets the speed to $R_{\min}$ by default and serves data of size B. Hence, at time $t_{J(\nu)}$ the queue size is at least $(J(\nu) + 1)B - B - (J(\nu) - 1)B/2 = (J(\nu) + 1)B/2$. ■

Corollary 3. *Suppose that the energy function has the form $f(s) = s^\alpha$ and that a rate adaptation policy* RA *uses energy at most $\nu \cdot \mathsf{opt}_B(t)$ by time t, for all times t. Then, the maximum queue size under* RA *is at least $\Omega\big(B \log_\nu(R_{\max}/R_{\min})\big)$.*

Sketch of proof: For $j = 1, 2, \ldots$, let

$$x_j = \frac{B}{R_{\min}} \left(2^\alpha \nu + 1\right)^{\frac{j-1}{1-\alpha}} ,$$

$$\text{and} \quad J(\nu) = \left\lfloor (\alpha - 1) \log_{2^\alpha \nu + 1} \frac{R_{\max}}{R_{\min}} + 1 \right\rfloor .$$

Algebraic manipulation shows that the above definition of x_i satisfies (3). ■

C. The Batch *policy*

We now show how to ensure both stability and near-optimal energy consumption using a rate adaptation policy that we call Batch. The basic idea is to wait until just enough data has arrived at the server so that our lower bound on $\mathsf{opt}_B(t)$ allows for a transmission rate that can serve all this data. Moreover, in the interest of simplifying the analysis, let us first assume that the server may operate even at rates higher than $R_{\max}$.

To begin with, define a *busy interval* as a time interval during which the queue is non-empty. Suppose that a busy interval I starts at time τ_0, and let $\tau_j = \min\{t \in I \mid A_e(\tau_0, t) \geq 2Bj\}$. Therefore, in each interval $[\tau_j, \tau_{j+1})$ data of size $2B$ arrives at the queue, and the policy makes sure that an equivalent amount of data is served by time $2\tau_{j+1} - \tau_j$. This is achieved by setting $r_e(t) = \max\left\{R_{\min}, \sum 2B/(\tau_{j+1} - \tau_j)\right\}$, where the summation is over all indices j such that $t \in [\tau_{j+1}, 2\tau_{j+1} - \tau_j)$. Denote these indices (if any such exist) by $j_1 < j_2 < \cdots < j_m$, and the interval $[\tau_{j_m}, \tau_{j_m+1})$ by $I(t)$. We have:

Lemma 4. *If $r_e(t) > R_{\min}$, then $r_e(t) \leq 6B/|I(t)|$.*

Proof: For a given t, note that $r_e(t) > R_{\min}$ guarantees the existence of indices $j_1, j_2, \ldots, j_m$ as defined above, for some $m \geq 1$, and thus also the existence of $I(t)$. Since $j_\ell \geq j_{\ell-1} + 1$, observe that $\tau_{j_\ell} \geq \tau_{j_{\ell-1}+1}$. Furthermore, $t \leq 2\tau_{j_\ell+1} - \tau_{j_\ell}$ implies $t - \tau_{j_\ell+1} \leq \tau_{j_\ell+1} - \tau_{j_\ell}$. Combining the above yields $t - \tau_{j_\ell+1} \leq \tau_{j_\ell+1} - \tau_{j_{\ell-1}+1}$, and hence $2(t - \tau_{j_\ell+1}) \leq t - \tau_{j_{\ell-1}+1}$. Consequently, $r_e(t)$ equals

$$\begin{aligned}
\sum_{\ell=1}^{m} \frac{2B}{\tau_{j_\ell+1} - \tau_{j_\ell}} &= \sum_{\ell=1}^{m-1} \frac{2B}{\tau_{j_\ell+1} - \tau_{j_\ell}} + \frac{2B}{\tau_{j_m+1} - \tau_{j_m}} \\
&\leq \sum_{\ell=1}^{m-1} \frac{2B}{t - \tau_{j_\ell+1}} + \frac{2B}{\tau_{j_m+1} - \tau_{j_m}} \\
&\leq \sum_{\ell=1}^{m-1} \frac{2^{\ell-m+1} 2B}{t - \tau_{j_{m-1}+1}} + \frac{2B}{\tau_{j_m+1} - \tau_{j_m}} \\
&\leq \frac{4B}{t - \tau_{j_{m-1}+1}} + \frac{2B}{\tau_{j_m+1} - \tau_{j_m}} \\
&\leq \frac{4B}{\tau_{j_m+1} - \tau_{j_m}} + \frac{2B}{\tau_{j_m+1} - \tau_{j_m}} \\
&= \frac{6B}{\tau_{j_m+1} - \tau_{j_m}} = \frac{6B}{|I(t)|} ,
\end{aligned}$$

where the last inequality is due to the fact that $t - \tau_{j_{m-1}+1} \geq t - \tau_{j_m} \geq \tau_{j_m+1} - \tau_{j_m}$. ■

Lemma 5. *The total energy used by* Batch *up until time t is at most $\sup_s \frac{f(6s)}{f(s)} \cdot \mathsf{opt}_B(t)$. For $f(s) = s^\alpha$, this is at most $6^\alpha \cdot \mathsf{opt}_B(t)$.*

Proof: Clearly, we need only consider the energy consumption of Batch during busy intervals – or, even more restrictively, during intervals where $r_e(t) > R_{\min}$. That holds because $R_{\min}$ is the most energy-efficient among the allowable rates, i.e. it minimizes the ratio $f(s)/s$, owing to our assumption about the energy function in Section I-A.

Within a busy period, consider some interval $[\tau_j, \tau_{j+1})$ and let $I'(\tau_j) = \{t \mid I(t) = [\tau_j, \tau_{j+1})\}$. Obviously $|I'(\tau_j)| \leq \tau_{j+1} - \tau_j$, by the definition of $I(t)$. Due to Lemmas 4 and 1, the energy consumption during $I'(\tau_j)$ is at most

$$\begin{aligned}
f\left(\frac{6B}{\tau_{j+1} - \tau_j}\right) \cdot I'(\tau_j) &\leq f\left(\frac{6B}{\tau_{j+1} - \tau_j}\right) \cdot (\tau_{j+1} - \tau_j) \\
&\leq \sup_s \frac{f(6s)}{f(s)} \cdot \mathsf{opt}_B(\tau_j, \tau_{j+1}) .
\end{aligned}$$

Repeating this for every interval of the form $[\tau_j, \tau_{j+1})$, across all busy periods, accounts for the energy consumption of all intervals where $r_e(t) > R_{\min}$, without overlaps. The lemma is thus established. ■

Subsequently, we derive a bound on the queue size.

Lemma 6. *The rate adaptation policy* Batch *guarantees that the queue size is never larger than $2B\big(\log_2(R_{\max}/R_{\min}) + 4\big)$.*

Sketch of proof: At time t, the amount of data still in the queue is given by

$$\sum_{\ell=1}^{m} 2B \cdot \frac{2\tau_{j_\ell+1} - \tau_{j_\ell} - t}{\tau_{j_\ell+1} - \tau_{j_\ell}} \le 2B \cdot m \ .$$

We need to bound m. On one hand, it is fairly straightforward to deduce that $t - \tau_{j_1+1} \le \tau_{j_1+1} - \tau_{j_1} \le 4B/R_{\min}$, otherwise the busy interval would end before τ_{j_1+1}. On the other hand, $t - \tau_{j_{m-1}+1} \ge \tau_{j_m+1} - \tau_{j_m} \ge B/R_{\max}$, because we assumed that $B \ge \sigma$. Together with $2(t-\tau_{j_\ell+1}) \le t-\tau_{j_{\ell-1}+1}$, the above yield $m \le \log_2(R_{\max}/R_{\min}) + 4$, as required. ∎

Finally, we explain how Batch works on a server whose maximum speed is strictly $R_{\max}$, which is the case of practical interest. More specifically, Batch tries to simulate its own behavior on an unrestricted server, which we described earlier. This involves two modes of operation, *normal* and *catch-up*. While in normal mode, the policy sets the same rate as it would set on the simulated unrestricted server. As soon as the latter rate exceeds $R_{\max}$, Batch enters catch-up mode, in which the rate is held at $R_{\max}$ constantly, and only reverts to normal mode when the queue sizes of the two servers, actual and simulated, coincide.

Naturally, in normal mode the bounds of Lemmas 5 and 6 remain valid. Suppose that just before switching to catch-up mode, the queue size is $Q \le 2B\big(\log_2(R_{\max}/R_{\min})+4\big)$. After remaining in that mode for a time interval of length λ, the queue size becomes $\le Q+\sigma+(1-\epsilon)R_{\max}\lambda-R_{\max}\lambda \le Q+\sigma$, which also implies that Batch will not stay in catch-up mode for ever. Furthermore, during the entire interval spent in catch-up mode, the actual server processed (at constant rate) exactly the same amount of data as the simulated one (at variable rate). Since $f(\cdot)$ is convex, the former server consumed no more energy than the latter.

Theorem 7. *The rate adaptation policy* Batch *ensures that maximum queue size is bounded by* $\Omega\big(B\log(R_{\max}/R_{\min})\big)$ *and consumes at most constant times the optimal energy.*

D. The SlowStart *policy*

In addition to Batch, let us introduce another policy named SlowStart. It stipulates that $r_e(t) = \min\{R_{\max}, g(t-\theta_e(t))\}$, where $g(\cdot)$ is a simple piecewise linear function

$$g(x) = \begin{cases} R_{\min} & \text{if } 0 \le x \le \frac{2B}{R_{\min}} \\ \frac{R_{\min}^2}{2B} \cdot x & \text{if } x > \frac{2B}{R_{\min}} \end{cases} ,$$

and $\theta_e(t)$ denotes the beginning of the busy interval of e that contains t, if one such exists, else is equal to t.

Within a busy interval, note that SlowStart initially sets the speed at $R_{\min}$ and later increases it linearly (hence the name), in a deterministic fashion. Interestingly, traffic arrivals have no direct effect on the speed, other than by determining when the busy interval ends. Below we summarize the properties of SlowStart in the single-server setting.

Theorem 8. *The rate adaptation policy* SlowStart *ensures that maximum queue size is bounded by* $\Omega\big(BR_{\max}^2/R_{\min}^2\big)$ *and consumes at most* $\sup_s \frac{f(2s)}{f(s)} \cdot \mathsf{opt}_B(t)$ *energy up until time* t*. For* $f(s) = s^\alpha$*, this is at most* $2^\alpha \cdot \mathsf{opt}_B(t)$*.*

Sketch of proof: Essentially a special case of Theorems 15 and 16 of Section IV, for $n = d = 1$. ∎

Remark. Although the above queue bound is worse than that of Batch, the SlowStart policy is nevertheless valuable, as we employ a variant of it in Section IV.

E. Queue-based policies

A queue-based policy sets the server rate according to $r_e(t) = r(q_e(t))$, where $q_e(t)$ is the queue size at time t, and $r(\cdot)$ is a non-negative and non-decreasing continuous function. In particular, if the queue size exceeds some threshold U then $r(q_e(t))$ is set to the maximum rate $R_{\max}$. Otherwise, $r(q_e(t))$ is at least the minimum rate $R_{\min}$. If traffic arrives at a constant rate, it is easy to provide good bounds for both queue and energy under such a policy.

Theorem 9. *Suppose that traffic arrives at the server at a constant rate* $\rho \le (1-\varepsilon)R_{\max}$*. Under a queue-based rate adaptation policy for which* $U = B$*, the queue size never exceeds* B *and the energy consumption up to time* t *is* $O(\mathsf{opt}_B(t))$*, for large enough* t*.*

Proof: Starting from an empty queue at time 0, both queue size and server speed begin to increase, until the speed reaches ρ. At that time, the queue size is at most U, because of the monotonicity of $r(\cdot)$ and the fact that $\rho < R_{\max}$. Both the server speed and queue size remain constant thereafter. Additionally, the energy usage up to time t is at most $t \cdot f(\rho)$, whereas $\mathsf{opt}_B(t) \ge t \cdot f\big((\rho t - B)/t\big)$. ∎

For arbitrary admissible traffic, it is easy to see that the above queue bound increases to $B + \sigma$, however we do not know whether any energy bound can be guaranteed.

III. Bounded Queue vs. Bounded Delay

We now turn to a network of multiple servers. Before diving into more complex results, let us state a simple but perhaps somewhat unexpected observation on the relationship between bounded queues and bounded delays. A folklore result states the equivalence between bounded queue and bounded delay when server rates are kept at $R_{\max}$.

Theorem 10 (Folklore)**.** *If a work-conserving algorithm always works at* $R_{\max}$*, then bounded delay is equivalent to bounded queue under any bounded adversary of rate* $(\sigma, 1-\varepsilon)$*.*

However, the above theorem no longer holds for some rate-adaptive algorithms. In particular, a stable rate-adaptive algorithm may not guarantee a bounded delay for every packet.

Lemma 11. *Neither* FTG *nor* NTS *guarantees a finite packet delay even in the permanent sessions model.*

Sketch of proof: Consider FTG on the following configuration. The network consists of three servers e_1, e_2, e_3 on a line and carries two connections: connection 1 goes to e_3 through e_1 and e_2, with rate $(1-2\varepsilon)R_{\max}$, and connection 2 goes to e_2 through e_1, with rate $\varepsilon R_{\max}$. Note that packets of connection 1 in the queue of e_1 have priority over those of connection 2, under FTG.

Suppose that at some point the queue of e_1 contains packets of both connections, and has size q° such that $r(q^\circ) \leq (1-2\varepsilon)R_{\max}$. Then, if connection-1 packets are henceforth injected at rate $r(q^\circ)$ and no more connection-2 packets are injected, existing connection-2 packets may be held *indefinitely* in e_1, even though its queue size remains bounded by q°. ■

IV. Temporary Sessions

As in Section II-E, consider a queue-based rate adaptation policy with threshold U. If $r(q_e(t)) < R_{\max}$ at time t, we say that all packets in the queue of server e are *withheld* in e at t. Naturally, a packet p may be withheld in e for two or more noncontiguous time intervals.

Lemma 12. *Denote by $w_e(t)$ the total size of packets in the queue of e at time t that are (or will be) withheld in e at* any *finite time $\geq t$. We have $w_e(t) \leq U$.*

Proof: To the contrary, suppose that $w_e(t) > U$ and let $t^* \geq t$ be the earliest time at which any of these packets is withheld in e. At that time, $q_e(t^*) > w_e(t) > U$, since none of said packets could have been processed until then. Consequently, $r(q_e(t^*)) = R_{\max}$ and hence no packet should be withheld at t^*, which is a contradiction. ■

Theorem 13. *Under any queue-based rate adaptation policy with finite threshold U,* FTG *and* NTS *are universally stable.*

Proof: Consider FTG; the proof for NTS is symmetric. We argue that having adversary $\mathcal{A}$ inject traffic in the rate-adaptive network G can be *quasi-simulated* by another adversary $\mathcal{A}'$ injecting traffic in a non-rate-adaptive system G', as described below. More precisely, we claim that it is within the capabilities of $\mathcal{A}'$ to ensure that at every time t the difference between the number of packets in each network is at most nU, where n is the number of servers of G.

The network G' is derived from G via the following construction: for each server e of G create d new servers $\pi_1(e), \pi_2(e), \ldots, \pi_d(e)$, and the resulting network is G'. All servers of G' follow the FTG algorithm as well. Furthermore, if $\mathcal{A}$ is a bounded adversary of rate $(\sigma, 1-\varepsilon)$, then $\mathcal{A}'$ is of rate $(\sigma + nU, 1-\varepsilon)$ and also knows the behavior of $\mathcal{A}$.

We now explain how the quasi-simulation takes place. Consider a packet p injected into G at time t, to be routed over consecutive servers $e_1, e_2, \ldots, e_k$, with $1 \leq k \leq d$. If p is not withheld anywhere until it is absorbed, then $\mathcal{A}'$ injects a copy of p into G' at t, with the same routing. Observe that $\mathcal{A}'$ can know what happens to p in the future, since the rate-adaptive network G is deterministic. On the other hand, suppose that p is withheld in $\ell \leq k$ distinct servers $e_{i_1}, e_{i_2}, \ldots, e_{i_\ell}$, such that $1 \leq i_1 < i_2 < \cdots < i_\ell \leq k$. For $1 \leq j \leq \ell$, denote by t_{i_j} the first time that p is withheld in e_{i_j} and by $\bar{t}_{i_j}$ the time that it is processed. In that case, $\mathcal{A}'$ injects a total of $\ell+1$ copies of p – not all at the same time – with mutually disjoint routing paths. The first copy is injected at t, with routing $e_1, \ldots, e_{i_1-1}, \pi_1(e_{i_1}), \ldots, \pi_{d-i_1+1}(e_{i_1})$. Subsequently, for $2 \leq j \leq \ell$, the j^{th} copy is injected at time $\bar{t}_{i_{j-1}}$, with routing $e_{i_{j-1}}, \ldots, e_{i_j-1}, \pi_1(e_{i_j}), \ldots, \pi_{d-i_j+1}(e_{i_j})$. Finally, the last copy is injected at time $\bar{t}_{i_\ell}$, with routing $e_{i_\ell}, \ldots, e_k$.

Recall that in the interval $[t_0, t)$, $\mathcal{A}$ is allowed to inject packets of total size up to $\sigma + (1-\varepsilon)R_{\max}(t-t_0)$ whose routing includes server e of G, whereas the analogous bound for $\mathcal{A}'$ is greater by nU. By Lemma 12, this suffices to allow $\mathcal{A}'$ to inject any subsequent copies of packets whose respective first copies were injected *before* time t_0. (Note that if only packets originally injected at or after t_0 had to be considered, a bound of $\sigma + (1-\varepsilon)R_{\max}(t-t_0)$ would be enough.) Additionally, it is straightforward to realize that at any time step t, the queue size of server e in G is exactly equal to $w_e(t)$ plus the size of the corresponding queue in G', which is bounded thanks to the universal stability of FTG [1]. ■

Remark. The quasi-simulation technique employed in the above proof cannot be used to obtain bounded delay guarantees for individual packets, thus it does not contradict Lemma 11.

A. *Combining* SlowStart *and* Batch *with* FTG

In order to combine the rate adaptation policy SlowStart with the scheduling algorithm FTG, we transform it so that it classifies packets in a similar way as FTG does, i.e. by the remaining distance to their destination. First, we provide a few definitions. Denote by $X_{e,i}(t)$ the total size of packets in the queue of server e at time t that are $\geq i$ hops away from their respective destinations. Moreover, define $k_i = 0$ for $i > d$ and $k_i = n\big(k_{i+1} + \sigma + 2R_{\max}^2(k_{i+1}+B)/R_{\min}^2\big)$ for $1 \leq i \leq d$.

For $t \geq 0$, the (revised) policy SlowStart sets the rate of e to $r_e(t) = \min\big\{R_{\max}, \max_{1\leq i\leq d} g_i\big(t - \theta_{e,i}(t)\big)\big\}$, where

$$g_i(x) = \begin{cases} R_{\min} & \text{if } 0 \leq x \leq \frac{2(k_{i+1}+B)}{R_{\min}} \\ \frac{R_{\min}^2}{2(k_{i+1}+B)} \cdot x & \text{if } x > \frac{2(k_{i+1}+B)}{R_{\min}} \end{cases}$$

and $\theta_{e,i}(t) = \sup\{t' \leq t \mid X_{e,i}(t') = 0\}$. Again, observe that each $g_i(\cdot)$ is a simple, piecewise linear function. Furthermore, the following property is readily verified:

Proposition 14. *For every $1 \leq i \leq d$ and every $x > 0$,*

$$g_i(x) = \max\left\{R_{\min}, 2\frac{\int_0^x g_i(u)\,\mathrm{d}u - k_{i+1} - B}{x}\right\}. \quad (4)$$

Theorem 15. *Combined with the rate adaptation policy* SlowStart, FTG *is universally stable and guarantees bounded packet delay.*

Proof: Via backwards induction on i, we argue that $\sum_e X_{e,i}(t) \leq k_i$ and

$$|X_{e,i}(t)| \leq k_{i+1} - \varepsilon R_{\max}\big(t - \theta_{e,i}(t)\big) + \sigma + \\ + 2R_{\max}^2(k_{i+1}+B)/R_{\min}^2 \quad (5)$$

for all $t \geq 0$, all $i \geq 1$, and every server e. These hold trivially for $i > d$, since the routing path of any packet contains $\leq d$

hops and thus $X_{e,i}(t)=0$. Now, suppose that both inequalities are also valid for $i=\eta+1$, where $1\le\eta\le d$. Then, for $i=\eta$, the packets in e at time t with $\ge\eta$ hops remaining may be partitioned in two categories: (i) packets that were injected at time $\theta_{e,\eta}(t)$ or later, whose total size is at most $\sigma+(1-\varepsilon)R_{\max}\big(t-\theta_{e,\eta}(t)\big)$, and (ii) packets that already existed in the system *before* $\theta_{e,\eta}(t)$, in queues *other than* that of e and still having $\ge\eta+1$ hops to go at that time, whose total size is at most $k_{\eta+1}$. Further, note that from $\theta_{e,\eta}(t)$ until t the queue processes at least $R_{\max}\big(t-\theta_{e,\eta}(t)-2R_{\max}(k_{\eta+1}+B)/R^2_{\min}\big)$ packets. Hence, $X_{e,\eta}(t)\le k_{\eta+1}-\varepsilon R_{\max}\big(t-\theta_{e,\eta}(t)\big)+\sigma+2R^2_{\max}(k_{\eta+1}+B)/R^2_{\min}$ and $\sum_e X_{e,\eta}(t)\le n\big(k_{\eta+1}+\sigma+2R^2_{\max}(k_{\eta+1}+B)/R^2_{\min}\big)=k_\eta$, completing the induction.

Moreover, (5) implies that $t-\theta_{e,i}(t)\le k_i/(n\varepsilon R_{\max})$. Therefore, a packet that arrives at the queue of e and still has $\ge i$ hops to go will be processed within $k_i/(n\varepsilon R_{\max})$ time. This means that every packet reaches its destination at most $\sum_{i=1}^d k_i/(n\varepsilon R_{\max})$ time after its injection. Finally, we deduce that the total number of packets in the system at any time is $\le k_1$. ■

Theorem 16. *Combined with the rate adaptation policy* SlowStart*,* FTG *consumes at most* $d\cdot\sup_s\frac{f(2s)}{f(s)}\cdot\mathsf{opt}_B(t)$ *energy up until time t, where d is the maximum hop count of routing paths. For* $f(s)=s^\alpha$*, this is at most* $d\cdot2^\alpha\cdot\mathsf{opt}_B(t)$*.*

Remark. In a multiple-server context, the meaning of $\mathsf{opt}_B(t)$ is slightly different than the one in Section II-A. Specifically, the desired bound B pertains to the sum of (i) the queue size of a server e at any given moment, and (ii) the total size of all packets that are elsewhere in the network at that time and must go through e in the future to arrive at their respective destinations. In all other respects, the statement and validity of Lemma 1 remain unaffected.

Proof of Theorem 16: Our analysis is done on a per-server basis. As usual, we need only account for energy usage in time intervals where $r_e(t)>R_{\min}$. First, define $\zeta_{e,i}(t)=\inf\{t'\ge t\mid X_{e,i}(t')=0\}$ and $I_i(t)=\big[\theta_{e,i}(t),\zeta_{e,i}(t)\big)$. Obviously $t\in I_i(t)$, and if $t_1\ne t_2$ then $I_i(t_1)$ and $I_i(t_2)$ either coincide or are mutually disjoint.

Furthermore, for $1\le i\le d$ let $T_i=\{t\in[0,t)\mid r_e(t)=g_i(t-\theta_{e,i}(t))>R_{\min}\}$ and $\mathcal{I}_i=\{I_i(t)\cap[0,t)\mid t\in T_i\}$. Now, fix i and consider each interval $I\in\mathcal{I}_i$. If $t_a=\inf I$ and $t_b=\sup I$, then we have

$$\begin{aligned}\int_{I\cap T_i}f(r_e(u))\,\mathrm{d}u&=\int_{I\cap T_i}f(g_i(u-t_a))\,\mathrm{d}u\\&\le f(g_i(t_b-t_a))\cdot|I\cap T_i|\\&\le f\left(2\frac{\int_{t_a}^{t_b}g_i(u)\,\mathrm{d}u-k_{i+1}-B}{t_b-t_a}\right)\cdot|I|\\&\le f\left(2\frac{A_e(t_a,t_b)-B}{t_b-t_a}\right)\cdot(t_b-t_a)\\&\le\sup_s\frac{f(2s)}{f(s)}\cdot\mathsf{opt}_B(t_a,t_b)\ ,\end{aligned}$$

where the first inequality above holds because $g_i(t-t_a)$ is non-decreasing in I, the second because of (4) plus the fact that $|I\cap T_i|\le|I|$, while the third is due to arguments similar to those that established (5). Since the intervals contained in each $\mathcal{I}_i$ are mutually disjoint, we straightforwardly deduce that the total energy consumed by e up until time t is at most

$$\begin{aligned}\sum_{i=1}^d\left(\int_{T_i}f(r_e(u))\,\mathrm{d}u\right)&=\sum_{i=1}^d\left(\sum_{I\in\mathcal{I}_i}\left(\int_{I\cap T_i}f(r_e(u))\,\mathrm{d}u\right)\right)\\&\le\sum_{i=1}^d\left(\sup_s\frac{f(2s)}{f(s)}\cdot\mathsf{opt}_B(t)\right)\ ,\end{aligned}$$

which directly implies the theorem. ■

Last but not least, it is rather straightforward to modify Batch along the same lines as we did with SlowStart, for the purpose of combining it with FTG and obtaining a stable rate-adaptive algorithm with near-optimal energy consumption. In fact, the queue bound thus derived is better than that of the SlowStart-FTG combination, consistently with the advantage of Batch over SlowStart in the single-server case. On the other hand, we could not derive a guarantee on end-to-end packet delay for the Batch-FTG algorithm, for reasons similar to those discussed in Section III.

Due to space constraints, we omit a detailed description and analysis of the aforementioned algorithm. Nevertheless, its salient properties are summed up in the following theorem.

Theorem 17. *Combined with the rate adaptation policy* Batch*,* FTG *is universally stable and consumes at most* $d\cdot\sup_s\frac{f(6s)}{f(s)}\cdot\mathsf{opt}_B(t)$ *energy up until time t. For* $f(s)=s^\alpha$*, this is at most* $d\cdot6^\alpha\cdot\mathsf{opt}_B(t)$*.*

V. Permanent Sessions

At this point, we turn our attention to the permanent sessions model, in which traffic is injected into fixed-path connections. Recall that σ_i is the burst size for connection i and ρ_i is the injection rate. Hence, if $H_i(t,t')$ is the amount of traffic injected into connection i during the time interval $[t,t')$, then $H_i(t,t')\le\sigma_i+\rho_i(t'-t)$. Let S_e be the set of connections passing through server e, let $q_{i,e}(t)$ be the amount of connection i data queued at server e at time t, and let $q_e(t)=\sum_{i\in S_e}q_{i,e}(t)$.

Consider a traditional setting in which each server e always runs at the maximum speed $R_{\max}$. The most commonly studied algorithm is Generalized Processor Sharing and its packetized version Weighted Fair Queueing (WFQ) [8], [14]. At all times GPS[2] splits its service equally among its backlogged sessions in proportion to the connection rates ρ_i. That is, at time t connection i is served at rate $R_{\max}\phi_i(t)$ where $\phi_i(t):=\rho_i\big/\big(\sum_{j\in S_e,q_{j,e}(t)>0}\rho_j\big)$. GPS is an idealized algorithm that assumes that service can be divided evenly among multiple sessions. For this reason Weighted Fair Queueing was introduced, which serves data on a packet-by-packet basis.

[2]Strictly speaking, the version of GPS called Rate Proportional Processor Sharing (RPPS).

WFQ runs a virtual GPS scheduler in the background and whenever it has to serve a packet it chooses the packet that would be scheduled next according to the GPS scheduler, assuming that no more packets arrive.

We now define rate-adaptive (RA) versions of GPS and WFQ that we call RA-GPS and RA-WFQ respectively. At all times t, server e operates at a speed $r(q_e(t))$. We focus on rate adaptation in the following generic form. If $q_e(t)$ exceeds a certain threshold U, then e operates at the full rate $R_{\max}$; otherwise, as long as the queue is non-empty, e operates at a rate no lower than the minimum rate $R_{\min}$. At time t any backlogged connection i is served at rate $r(q_e(t))\phi_i(t)$. RA-WFQ operates an RA-GPS scheduler in the background and always runs the server at the same speed as RA-GPS. Whenever it has to serve a packet it chooses the packet that would be scheduled next according to the RA-GPS scheduler, assuming that no more packets arrive.

Note that even though $q_{i,e}(t)$ may have different values depending on whether we use RA-GPS or RA-WFQ, both the queue size $q_e(t)$ and the server processing rate $r(q_e(t))$ are independent of whichever of the two algorithms is used.

The classic analysis of Parekh and Gallager [14] showed that if $r(q) = R_{\max}$ for all q then the end-to-end delay bound for connection i under WFQ is given by,

$$\frac{\sigma_i + (K_i - 1)L_i}{\rho_i} + K_i \frac{L_{\max}}{R_{\max}} \ , \tag{6}$$

where K_i is the number of hops on the path of connection i, L_i is the size of the maximum packet injected into connection i, and $L_{\max} = \max_i L_i$. The aim of this section is derive a similar bound for RA-WFQ. Our analysis in motivated by a derivation of the bound (6) presented in the book [6]. In particular, we show:

Theorem 18. *The end-to-end delay for connection-i packets under* RA-WFQ *is bounded by,*

$$\frac{\sigma_i + (K_i - 1)L_i}{\rho_i} + K_i \left(\frac{L_{\max}}{R_{\min}} + \frac{R_{\max}}{R_{\min}} \frac{U}{\rho_i} \right) .$$

Proof: Let us first concentrate on connection i at one particular server e. For the nth packet arrival from connection i at link e, let a_n be its arrival time at e, and let ℓ_n be its packet size. By the definition of GPS, ρ_i is a lower bound on the service rate to connection i. (Note that $\sum_{j \in S_e} \rho_i \le R_{\max}$ by assumption.) Moreover, define a sequence

$$h_n = \max\{a_n, h_{n-1}\} + \frac{\ell_n}{\rho_i} \ .$$

Following [6], we say that a server is a (ρ_i, y) Guaranteed Rate ((ρ_i, y)-GR) server if the finishing time of the nth packet is at most $h_n + y$ for all n. It is shown in [6] that under GPS server e is a $(\rho_i, 0)$-GR server for connection i.

Let g_n^{RA} be the finishing time for the nth packet under RA-GPS. In the following, we bound the difference between h_n and g_n^{RA}.

Lemma 19. $g_n^{\mathsf{RA}} - h_n \le \frac{R_{\max}}{R_{\min}} \frac{U}{\rho_i}$. *Hence, under* RA-GPS *server e is a $\left(\rho_i, \frac{R_{\max}}{R_{\min}} \frac{U}{\rho_i}\right)$-GR server for connection i.*

Proof: It is not hard to see that if connection i is always served at rate exactly ρ_i then the finishing time of the nth packet will be exactly h_n. We examine $s_i(t)$, the cumulative amount of service connection i receives by time t if it is always offered server ρ_i. The slope of $s_i(t)$ is either ρ_i or 0. (The latter happens when the connection-i queue is empty.) We define $s_i^{\mathsf{RA}}(t)$ to be the cumulative service connection i receives under RA-GPS. We now claim that $s_i(t) - s_i^{\mathsf{RA}}(t) \le U$ for any t. This is because whenever $s_i(t) - s_i^{\mathsf{RA}}(t)$ reaches U it must be the case the value of $q_{i,e}(t)$ under RA-GPS is at least U. This in turn implies that $q_e(t) \ge U$ and so $r(q_e(t)) = R_{\max}$. Hence RA-GPS serves connection i at rate at least ρ_i and so $s_i(t) - s_i^{\mathsf{RA}}(t)$ cannot increase.

The above argument implies that $s_i(h_n) - s_i^{\mathsf{RA}}(h_n) \le U$. Since connection-$i$ is always served at rate at least $R_{\min}\rho_i / R_{\max}$, even under RA-GPS, this in turn implies that $g_n^{\mathsf{RA}} \le h_n + \frac{R_{\max} U}{R_{\min} \rho_i}$, as required. ■

We now bound the difference in finishing time for a packet at server e under RA-GPS and RA-WFQ. The intuition here is that processing a connection-i packet under RA-WFQ can be delayed if a packet from a different connection just started being processed. The worst-case delay happens when the other packet is large in size and is processed at the lowest possible rate.

Lemma 20. *The finish time for a packet under* RA-WFQ *is at most its finish time under* RA-GPS *plus* $\frac{L_{\max}}{R_{\min}}$.

Sketch of proof: Entirely analogous to the corresponding result for WFQ in [6]. Omitted here for reasons of space. ■

Corollary 21. *Under* RA-WFQ*, server e is a $\left(\rho_i, \frac{R_{\max} U}{R_{\min} \rho_i} + \frac{L_{\max}}{R_{\min}}\right)$-GR server for connection i.*

Subsequently, we focus on concatenating a sequence of GR links, using a concatenation theorem from [6].

Theorem 22 (Concatenation Theorem [6])**.** *The concatenation of M (x_j, y_j)-GR servers, where $1 \le j \le M$, results in an (x, y)-GR server where $x = \min_{1 \le j \le M} x_j$ and $y = \sum_{1 \le j \le M-1} (y_j + L_i / x_j)$.*

Suppose connection i goes through K_i links. We know that each link e along the path is (x_e, y_e)-GR, where x_e and y_e are defined by Corollary 21. Applying Theorem 22 to Corollary 21 yields:

Lemma 23. *For the K_i links along the path for connection i, concatenating these K_i links results in an (x_i, y_i)-GR server, where $x_i = \rho_i$ and $y_i = K_i \big(\frac{R_{\max} U}{R_{\min} \rho_i} + \frac{L_{\max}}{R_{\min}}\big) + (K_i - 1)\frac{L_i}{\rho_i}$.*

Next, we appeal to the following result from [6] to obtain an end-to-end delay bound for RA-WFQ.

Lemma 24 (Delay Bound [6])**.** *For a (σ, ρ) traffic source, a (ρ, y)-GR server guarantees a delay of $\frac{\sigma}{\rho} + y$.*

Combining Lemmas 23 and 24, we derive that the end-to-end delay for connection i under RA-WFQ is upper bounded

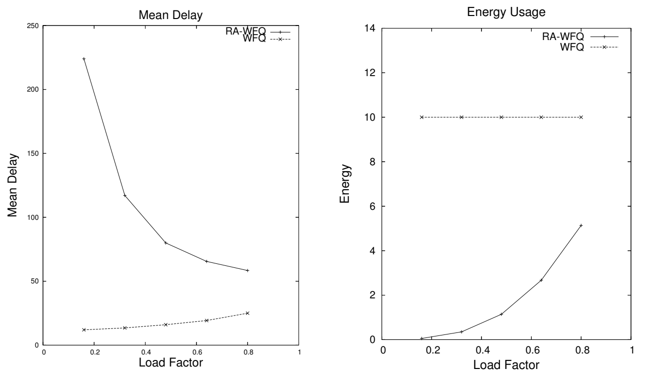

Fig. 1. The delay and energy performance of WFQ and RA-WFQ.

by

$$\frac{\sigma_i + (K_i - 1)L_i}{\rho_i} + K_i\left(\frac{L_{\max}}{R_{\min}} + \frac{R_{\max}}{R_{\min}}\frac{U}{\rho_i}\right),$$

as required. ∎

A. Simulation

We now briefly describe simulation results that illustrate the tradeoff between delay and energy usage. For our experiments, we use a home-grown simulator written in C. The network topology is a linear array of 10 identical servers. Moreover, the network supports one long connection passing through all 10 servers, and 10 short connections each passing through one of the servers. We normalize the maximum processing rate $R_{\max}$ of each server to 1, and consider the energy function $f(s) = s^3$. The injection rate of the long connection is $\ell/4$, for some load factor ℓ that we vary between 0.16 and 0.8. The injection rate of each short connection is $3\ell/4$. Hence, the load on each server is ℓ. Observe that our study is explicitly scale-invariant: time is specified in time slots, injection and processing rates are specified as a fraction of $R_{\max}$, and power consumption is specified as a function of the processing rate.

The left-hand plot of Figure 1 shows the average delay of packets under WFQ and RA-WFQ, where the latter uses the queue-based rate adaptation policy given by $r(q) = \min\{q/10, 1\}$. Similarly, the right-hand plot shows energy usage. As expected, RA-WFQ exhibits significantly lower energy consumption for light loads, at the expense of higher packet delays.

VI. Conclusion

In this paper we studied energy-aware scheduling algorithms with the objective of achieving bounded queue sizes (which implies network stability) and bounded delays, while simultaneously minimizing energy usage. We proposed several policies, with varying degrees of complexity, to adjust the server rates in response to incoming traffic. By combining these policies with existing stable algorithms, such as Farthest-to-Go and Weighted Fair Queueing, we were able to achieve the aforementioned objective and also derive different tradeoffs between queue sizes/delays and energy consumption.

Several interesting open questions remain. For example, consider Longest-in-System (LIS), which gives priority to packets that have been in the network for the longest time; thus, it may be regarded as a "global" FIFO. Assuming that servers run at $R_{\max}$ all the time, LIS is known to be universally stable [1] and has many desirable properties. Consequently, inventing a rate-adaptive version of LIS with near-optimal energy usage, while also preserving its salient characteristics, would be a substantial achievement.

Finally, on a more general level, it would be very interesting to know whether there is a single rate-adaptive approach that can be applied to *any* stable scheduling algorithm to guarantee bounded queue size and near-optimal energy consumption.

Acknowledgement: This R&D work was completed with the support of the U.S. Department of Energy (DOE), award no. DE-EE0002887. However, any opinions, findings, conclusions and recommendations expressed herein are those of the authors and do not necessarily reflect the views of the DOE.

References

[1] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results and performance bounds for greedy contention-resolution protocols. *Journal of the ACM*, 48(1):39–69, January 2001.

[2] M. Andrews, A. Fernandez Anta, L. Zhang, and W. Zhao. Routing and scheduling for energy and delay minimization in the powerdown model. In *IEEE INFOCOM '10*, March 2010.

[3] M. Andrews and L. Zhang. The effects of temporary sessions on network performance. In *SODA '00: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457, San Francisco, CA, USA, January 2000.

[4] N. Bansal, H.-L. Chan, and K. Pruhs. Speed scaling with an arbitrary power function. In *SODA '09: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 693–701, Philadelphia, PA, USA, 2009.

[5] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM*, 54(1), 2007.

[6] J.-Y. Le Boudec and P. Thiran. *Network Calculus*. Springer Verlag, http://ica1www.epfl.ch/PS_files/NetCal.htm, 2004.

[7] H.-L. Chan, W.-T. Chan, T. W. Lam, L.-K. Lee, K.-S. Mak, and P. W. H. Wong. Energy efficient online deadline scheduling. In N. Bansal, K. Pruhs, and C. Stein, editors, *SODA '07*, pages 795–804, 2007.

[8] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking: Research and Experience*, 1:3–26, 1990.

[9] Enhanced Intel SpeedStep technology for the Intel Pentium M processor. Intel White Paper 301170-001, 2004.

[10] M. Gupta and S. Singh. Greening of the Internet. In *SIGCOMM '03*, pages 19–26, New York, NY, USA, 2003.

[11] S. Irani and K. Pruhs. Algorithmic problems in power management. *SIGACT News*, 36(2):63–76, 2005.

[12] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In J. Crowcroft and M. Dahlin, editors, *NSDI '08*, pages 323–336, 2008.

[13] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, 1993.

[14] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple-node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, 1994.

[15] A. Wierman, L. L. H. Andrew, and A. Tang. Power-aware speed scaling in processor sharing systems. In *IEEE INFOCOM '09*, pages 2007–2015, 2009.

[16] F. F. Yao, A. J. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *FOCS '95*, pages 374–382, 1995.