Green Communications and Networking

April 12, 2012

Contents

Chapter 1

Architectural Design of An Energy-efficient Router

[*]Chengchen Hu and [+]Bin Liu and [†]Mingui Zhang and [‡]Beichuan Zhang and [§]Xiaojun Wang
[*]MoE KLINNS, Department of Computer Science and Technology, Xi'an Jiaotong University, chengchenhu@xjtu.edu.cn
[+]Department of Computer Science and Technology, Tsinghua University, liub@tsnghua.edu.cn
[†]Huawei Inc., zhangmingui@huawei.com
[‡]Department of Computer Science, Arizona University, bzhang@arizona.edu
[§]School of Electronic Engineering, Dublin City University, xiaojun.wang@dcu.ie

The tremendous success of Internet makes it become a ubiquitous infrastructure nowadays comprising an enormous number of hardware components to delivery a variety of services to end users. However, the concern of energy consumption in today's Internet infrastructure is highlighted in [2]: in 2007, the annual energy consumption of the Internet terminal devices and networking and transmission equipment is about 350 billion kWh in the US (9.4% of the national electricity consumption), and about 868 billion kWh globally (5.3% of the global electricity consumption). In the most recent years, with the aim of building a more sustainable society, research efforts have been made to look into the feasibility and benefits of applying energy efficient techniques in Information and Communication Technology (ICT) systems in order to obtain the best tradeoff between energy cost and system performance.

In fact, many power management solutions are available for years and have been well applied to Personal Computer (PC) systems and different types of battery-operated portable devices, e.g., dynamic voltage scaling technology in Central Processing Unit (CPU) of portable devices, energy-saving sleep modes in PC operating systems, and power-aware routing protocols in Wireless Sensor Networks (WSNs). However, the application of these techniques in the Internet-scale systems has not been fully considered yet. Some pioneering exploitations of saving energy across the Internet are discussed [8] which investigated the network protocols design with energy-saving considerations and device power consumption optimization techniques against expected performance. The previous studies have confirmed the feasibility and benefits of engineering the next generation energy-saving Internet infrastructure and pointed out a set of research directions [8, 1, 10]. In this chapter, we concentrate on the exploration of power/energy-saving mechanisms through the design of Internet transmission equipment, e.g., routers. By revisiting the characteristics of the Internet behaviors and the modular architecture of routers, this chapter suggests the approach for engineering energy efficient Internet from three different perspectives and discusses the imposed technical challenges. To address the challenges and seize the energy-saving opportunities, a new conceptual router model/architecture as the guide to design and implement power efficient router, as well as the Internet, is pursued.

1.1 Opportunities and Challenges

Several intrinsic characteristics of Internet behaviors imply the opportunities of applying energy-saving techniques in routers towards the energy-efficient Internet.

Low average link utilization and significant path redundancy. Over-provisioning of the link bandwidth is common in network planning to harness burst traffic. It is stated that the average utilization of backbone links is less than 30% [18]. Besides, massive link redundancies are also built to survive the network during potential failures. Although low link utilization and massive link redundancy improve the network resilience, they greatly hurt the energy efficiency of Internet. The links are operated at full rate all the time while they are highly under-utilized most of the time. Our first opportunity to save energy is shifting and aggregating traffic from lightly utilized links. In this way, some of the routers (or line-cards in a router) can be shut down to reduce the power instead of in operation all the time.

Variable Internet traffic demand. It is now well known that the Internet traffic demand exhibits fluctuations over time due to end user behavior, temporary link failures, anomalies and so forth [5]. The dynamics of data traffic rate allows us to incorporate energy control mechanisms in the router design, e.g., adapting router processing speed based on the detected traffic rate or queue length. The processors can be decelerated to reduce the energy consumption for incoming traffic with low data rate.

Energy-efficient packet processing. At present, the packet processing mechanisms inside a router are mainly driven by the pursuit of desired performance, with little consideration of energy consumption. From the energy efficiency perspective, there remains a large potential to optimize the design of router packet processing components through various of techniques, e.g., reducing the peak power of the processor using architectural approaches.

Although the aforementioned opportunities are promising to minimize the energy consumption in current Internet infrastructure, they also impose us with the following questions:

How to properly aggregate traffic and route packets? With traffic aggregation and shift in mind, the end-to-end routing paths of individual flows needs to be determined which are not necessarily following the conventional shortest path first routing paradigm, e.g., Open Shortest Path First (OSPF) protocol. Mathematically, it can be formulated as an optimization problem, which maximizes the energy-savings by identifying as many idle links as possible to be put into sleep mode whilst guaranteeing the expected network performance. The nature of Internet traffic dynamics over time, e.g., variable traffic demand, imposes additional complexity on finding the solution and needs to be taken into account.

How to manage routers under energy saving states to achieve performance-energy tradeoff? Generally, off-the-shelf routers can only work in one of two operational states: on and off, which can not be flexibly configured or switched to cope with traffic fluctuations. To meet the needs of energy consumption reduction, we suggest that multiple router operational states (e.g., energy saving state) with fast switching ability among them should be supported. Advanced strategies determining when to switch and to which state should be adopted to achieve the best tradeoff between the performance and energy efficiency.

How to reduce peak power with performance guarantees? It is also quite challenging to reduce the router's peak power without harming the system performance in the worst case. Even the router could freely switch working states to reduce the average power, it is impossible to lower the peak power, which is related to the processing in the worst case, without changes on the router architecture. Unique features of network processing should be utilized and elaborate architectural design of router functions should be investigated to take this challenging task.

1.2 Architecture of Green Reconfigurable Router

The concept of Green Reconfigurable Router (i.e., GRecRouter) is presented in [10], which aims to contribute to the creation of the next generation energy-efficient Internet infrastructure. Through the enhancement of the router architectural design, it is expected to reduce both average power and peak power consumption during network operation.

GRecRouter is designed in such a way that its settings (e.g., routing path, clock frequency, supply voltage) are "reconfigurable" based on its awareness of traffic rate fluctuations. In details, this can be interpreted from two aspects as follows.

- At a large time-scale, it has been known for a long time that the Internet traffic exhibits strong daily and weekly patterns and the behavior remains unchanged over years [5]. Take the enterprise network as an example, the traffic volume in daytime could be about ten times more than that in night time, and the similar difference is also observed between workdays and weekends. The time-of-day/time-of-week effect makes the link load varies slowly, but with a large magnitude. Bear this in mind, GRecRouter firstly manages the energy consumption of the Internet in a macro time scale by periodically aggregating traffic during lightly loaded periods, and in contrast distributing traffic in heavily loaded periods. The realization of this control mechanism needs modifications of the underlying routing protocols and the mechanism of routing path selection.

- At a small time-scale, the flow rate varies in a smaller magnitude but more frequently (compared with time-of-day/time-of-week effect). According to this, GRecRouter also adopts an energy control mechanism operating in a micro time scale, which adaptively tunes the processing rate of the function blocks inside individual routers based on the detected link utilization, traffic rate or queue length. To minimize the energy consumption, fast but more energy-consuming states are suggested to be used under heavy traffic scenarios, while slow but energy efficient states are suitable for light traffic scenarios.

With the elaborate design of functional blocks inside the GRecRouter using architectural advances, the router's peak power could also be reduced. The energy-efficient architectural designs for implementing main router functions in GRecRouter, i.e., routing lookup and packet queuing, are presented in later sections.

To summarize, the design of GRecRouter exhibits many desirable and unique features compared with conventional routers, as shown in Fig. 1.1. The major features are discussed as follows:

- At the network level, power-aware routing is applied to determine the end-to-end routing paths and forward the packets from the source to the destination with minimal energy consumption. It may change packet routes during quite periods for traffic aggregation and inform the idle devices to work in sleep mode. It should be noted that the expected performance, e.g., Quality of Service (QoS), needs to be guaranteed with the energy-saving consideration.
- At the node level, rate adaptive processing should be activated inside individual routers.

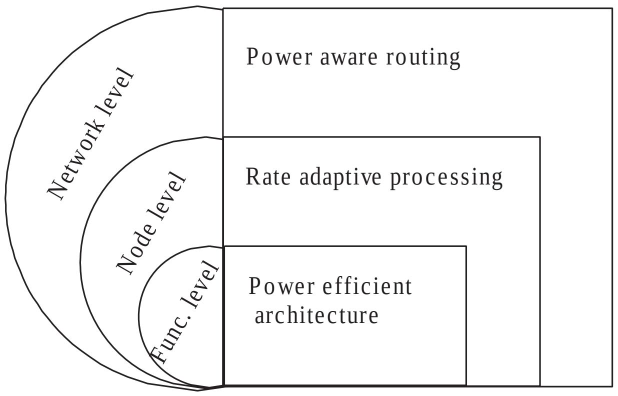


Figure 1.1: GRecRouter controls power dissipation in three levels.

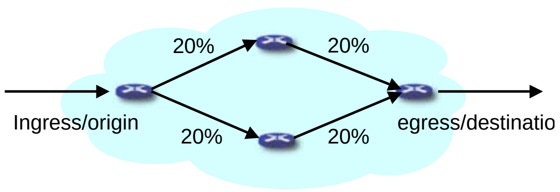
Different function blocks could be flexibly configured to be operated in a specific state, e.g. on, off or low energy-consumption states (with slower clock frequency or/and lower supply voltage) according to the network traffic load.

- At the function block level, each functional block should be designed with energy-efficiency in mind so as to reduce the peak power of the router. Many architectural design techniques could be employed as appropriate, e.g., caching, clock gating, and processing separation.

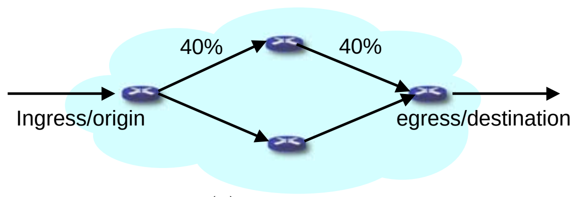
The GRecRouter is presented as a conceptual architectural model which is not limited to any specific deployment. The analysis and discussions on a reference implementation of GRecRouter based routers are presented in details in the following sections.

1.3 Power Aware Routing through Green Traffic Engineering

Today's wide-area networks usually have redundant and over-provisioned links, resulting in low link utilization most of the time. Take Abilene, a large US education backbone, as an example. It is shown that the average link utilization is only about 2%, the maximum fluctuates mostly between 10% and 20%, and only one rare event pushes the maximum over 50% [20]. High path redundancy and low link utilization combined also provide a unique opportunity for power-aware traffic engineering. Traditional traffic engineering spreads the



(a) traditional



(b) power-aware

Figure 1.2: Different traffic engineering goals

traffic evenly in a network (Figure 1.2(a)), trying to minimize the chance of congestion induced by traffic bursts. However, in power-aware traffic engineering (Figure 1.2(b)), one can free some links by moving their traffic onto other links, so that the links without traffic can go asleep for an extended period of time. This should result in more power saving than pure opportunistic link sleeping because the sleep mode is much less likely to be interrupted by traffic.

To generalize the basic idea illustrated in Figure 1.2(b), we develop the GreenTE model, which, given the network topology and traffic matrix, finds a routing solution (i.e., the links to be used and the traffic volume to be carried on each link) that maximizes the power saving from turning off line-cards as well as satisfying performance constraints including link utilization and packet delay.

1.3.1 The General Problem Formulation

We model the network as a directed graph $G = (V, E)$, where V is the set of nodes (i.e., routers) and E is the set of links. A port can be put to sleep if there is no traffic on the link, and a line-card can be put to sleep if all its ports are asleep. Let M be the set of line-cards in the network. For a single line-card $m \in M$, its base power consumption is B_m, its set of ports is S_m, and each port $l \in S_m$ consumes power P_l, then the power saving from turning of one port is P_l, and the power saving from turning off the entire line-card is $B_m + \sum_{l \in S_m} P_l$. The objective is to find a routing that maximizes the total power saving in the network. This general power-aware traffic engineering problem can be formulated based on the Multi-Commodity Flow model as follows. Please see Table 1.1 for the notation used in this chapter.

Table 1.1: Summary of notation used in this chapter

Notation	**Meaning**
S_m	Set of links connected to line-card m
P_l	Power consumption of the port connected to link l
B_m	Base power consumption of line-card m
x_l	1 if link l is sleeping, 0 otherwise
y_m	1 if line-card m is sleeping, 0 otherwise
$f_l^{s,t}$	Traffic demand from s to t that is routed through link l
H_l	l's head node
T_l	l's tail node
I_l^v	1 if v is the head node of link l, 0 otherwise
O_l^v	1 if v is the tail node of link l, 0 otherwise
$D_{s,t}$	Traffic demand from s to t
C_l	Capacity of link l
u_l	Utilization of link l
$r(l)$	Reverse link of l
k	Number of candidate paths for each origin-destination (OD) pair
U_T	Threshold for the maximum link utilization (MLU)
$Q_i^{s,t}(l)$	1 if the ith candidate path from s to t contains link l, 0 otherwise
$\alpha_i^{s,t}$	Ratio of traffic demand from s to t that is routed through the ith candidate path

$$\text{maximize} \quad \sum_{l\in E} P_l x_l + \sum_{m\in M} B_m y_m \tag{1.1}$$

$$\text{s.t.} \quad \sum_{l\in E} f_l^{s,t} O_l^i - \sum_{l\in E} f_l^{s,t} I_l^i =$$

$$\begin{cases} D_{s,t}, & i = t \\ -D_{s,t}, & i = s \\ 0, & i \neq s,t \end{cases}, \quad s,t,i \in V, s \neq t \tag{1.2}$$

$$|S_m|\, y_m \leq \sum_{l\in S_m} x_l \tag{1.3}$$

$$u_l = \frac{1}{C_l} \sum_{s,t\in V, s\neq t} f_l^{s,t}, \quad l \in E \tag{1.4}$$

$$x_l = x_{r(l)}, \quad l \in E \tag{1.5}$$

$$x_l + u_l \leq 1, \quad l \in E \tag{1.6}$$

Equation 1.1 computes the objective that maximizes the total power saving in the network. Equation 1.2 states the flow conservation constraints. Let $|S_m|$ be the cardinality of S_m, then equation 1.3 ensures that a line-card is put to sleep only when all its ports are asleep. Equation 1.4 calculates the link utilization. Equation 1.5 ensures that links are put to sleep in pairs, i.e., there is no inbound traffic nor outbound traffic. Equation 1.6 states that a link can be put to sleep only if there is no traffic on it, and when it is on, it does not carry traffic more than its capacity. Solving this problem gives which links to be turned off, and how much traffic each remaining link should carry.

The binary (integer) variables x_l and y_m that denote the power state of link l and line-card m make the model a Mixed Integer Programming (MIP) problem. Generally speaking, MIP problems are NP-Hard, thus its computation time for medium and large networks is a concern. This model, though it maximizes power saving in the network, does not consider some practical constraints. For example, packet delay could be much longer than that of current shortest path routing, and links may operate at unacceptably high link utilization, making them vulnerable to any traffic bursts.

1.3.2 A Practical Heuristic

To consider the practical constraints and reduce computation time, we refine the problem formulation as follows.

$$\text{maximize} \tag{1.7}$$

$$\sum_{l \in E} P_l x_l + \sum_{m \in M} B_m y_m \tag{1.8}$$

$$\text{s.t.} \tag{1.9}$$

$$f_l^{s,t} = \sum_{0 \le i < k} Q_i^{s,t}(l) D_{s,t} \alpha_i^{s,t}, s, t \in V, l \in E, s \neq t \tag{1.10}$$

$$\sum_{0 \le i < k} \alpha_i^{s,t} = 1, \quad s, t \in V, s \neq t \tag{1.11}$$

$$|S_m|\, y_m \le \sum_{l \in S_m} x_l \tag{1.12}$$

$$u_l = \frac{1}{C_l} \sum_{s,t \in V, s \neq t} f_l^{s,t}, \quad l \in E \tag{1.13}$$

$$x_l = x_{r(l)}, \quad l \in E \tag{1.14}$$

$$x_l + u_l \le 1, \quad l \in E \tag{1.15}$$

$$u_l \le U_T, \quad l \in E \tag{1.16}$$

One change is the addition of the bound on maximum link utilization (MLU) in a network. Equation 1.16 states that MLU must be no greater than a configured threshold U_T. In this chapter, we use 50% as the default value of U_T.

Another change is the use of the k-shortest paths as the candidate paths instead of searching the solution in all possible paths. Equation 1.10 and 1.11 are equivalent to the flow conservation constraints under this change. It reduces overall computation time as well as adding path length as another constraint. The general model introduced in the previous subsection considers all possible paths for each origin-destination (OD) pair, making the search space extremely large. To reduce search space and computation time, for each OD pair, we pre-compute its set of k-shortest paths and only search solutions within this set. Since the k-shortest paths are pre-computed with network topology as the only input, they do not change with the traffic matrix and the computation does not add run-time overhead. Note that when k is set to be large enough, we can actually consider all possible paths for each OD pair, which will give the maximal power saving under the MLU constraint. However, the computation time increases with the value of k; therefore there is a trade off between the precision of the heuristic and the computation time. Our evaluation later will show that a reasonably large k can achieve near optimal results.

Searching solutions only within k-shortest paths also avoids very long paths. In practice, network operators can have their own definitions of link delays and path lengths, and choose the set of k candidate paths accordingly. In this chapter we add up link propagation delays to get path lengths, and consider two different constraints in selecting the k-shortest path.

One is that any candidate path should not be longer than the diameter of the network, i.e., the shortest path between the farthest pair of nodes in the topology. The other is that between any OD pair, the path length of the candidate should not be greater than twice that of the shortest path. Depending on how the candidate paths are chosen, in this chapter, we will evaluate three different combinations:

- *basic*: The candidate paths are the k-shortest paths. MLU bound is applied.
- *basic+nd*: The candidate paths are k-shortest paths which also satisfy the network diameter constraint. MLU bound is applied.
- *basic+e2e*: The candidate paths are k-shortest paths which also conform to the OD-pair end-to-end delay constraint. MLU bound is applied.

With these changes, the GreenTE model now has practical constraints on link utilization and path length, and also can be solved within reasonable time.

Figure 1.3 shows that the power saving potential grows as the value of k increases. However, increasing k also increases the computation time. When k is large enough (20 in this example), increasing k only improves the power saving potential by a small amount. Therefore, GreenTE is able to achieve near optimal power savings as long as k is reasonably large.

As mentioned previously, the computation time becomes unacceptable for large topologies such as Sprint and AT&T. We use CPLEX [12] as the computation tool and in order to limit the computation time, we force CPLEX to stop after 300 seconds. Table 1.2 shows that the computation time for near optimal results within the k-shortest paths solution space increases dramatically as value of k grows. When $k = 20$, we can obtain about 96% of the optimal power saving potential when we limit the computation time to be 300 seconds. More details about GreenTE can be found in [20].

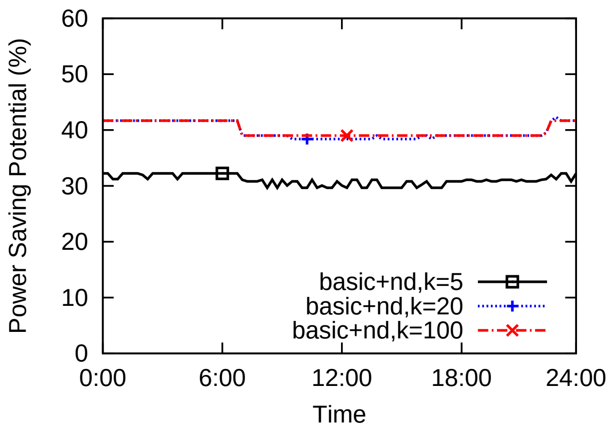


Figure 1.3: Power saving potential of GÉANT with different k values.

Table 1.2: Power saving potential of AT&T under *basic+e2e* with traffic that has 21% MLU under OSPF

k value	Computation Time	Status	Power Saving Potential
5	65s	Optimal	11.90%
10	5747s	Optimal	17.54%
20	100892s	Optimal	19.79%
20	300s	Non-optimal	18.99%

1.4 Rate Adaptive Processing inside Routers

1.4.1 Dynamic Voltage and Frequency Scaling

Amongst the factors determining the dynamic power, voltage and frequency have a far-reaching impact. The dynamic power is proportional to the frequency and the square of supply voltage. The frequency also influences the supply voltage: maintaining a higher clock frequency may mean maintaining a higher supply voltage. Therefore, it has a cubic impact on power dissipation combining the voltage and frequency. Although voltage and frequency have significant leverage on energy savings, it may degrade the system performance due to the decrease of clock frequency. If one could recognize the periods when absolute guarantees or stringent requirements of network performance are not required, the energy can be greatly saved by reducing the voltage and frequency. Power-aware technique managing the supply voltage and clock frequency, known as Dynamic Voltage and Frequency Scaling (DVFS) has been extensively studied in the area of microprocessor (e.g., CPU) design. However, little research work has been reported on the application of DVFS technique in the context of router design.

Consider a router operation scenario: a router completes the processing of all the packets within a time interval, and the energy will be wasted in the remaining idle time periods. A coarse granularity DVFS method is proposed in [7] for Ethernet devices, which attempts to identify the inactive periods of the links and puts the associated devices into sleep in such periods. As shown in Fig. 1.4 (a), if the processing could be completed by the middle of the interval, half of the energy could be saved in an ideal case[1]. The key question behind this approach is in deciding when to turn off a link and for how long. Without the detailed knowledge of incoming traffic pattern, this approach has to compromise with network performance (e.g., packet delay, loss) on energy consumption reduction.

Rather than simply pausing the packet processing in idle periods, more energy is expected to be saved if the packet processing is stretched to a time interval by scaling the frequency and voltage, as illustrated in Fig. 1.4 (b) which is with the same traffic load as in Fig. 1.4 (a). It reduces the clock frequency by half and stretches the processing time covering the whole time interval. Meanwhile, with halved clock frequency, the voltage could also be scaled down. Suppose that the voltage could also be halved, then the total energy needed is reduced to 1/8 of its normal consumption considering the quadratic impact of voltage on power dissipation. It is obvious that more gain will be achieved if stretching the processing into a larger interval, yet resulting in a longer packet delay. Therefore, the energy saving can be optimized subject to the constraint of maximum tolerable delay.

1.4.2 Adaptive Link Rate Interface

Take Ethernet as an example, a network 1 Gbps interface consumes 4W more than a 100Mbps interface and a 10Gbps interface consumes about 10 to 20 W. In addition, the impact of link utilization on the power consumption is considered minimal [6]. Therefore, the interface data rate should be dynamically tailored with flexible configuration according to the link utilization. This approach is known as Adaptive Link Rate (ALR), which was proposed in

[1]Suppose zero switching time to sleep state and zero energy consumption in sleep state.

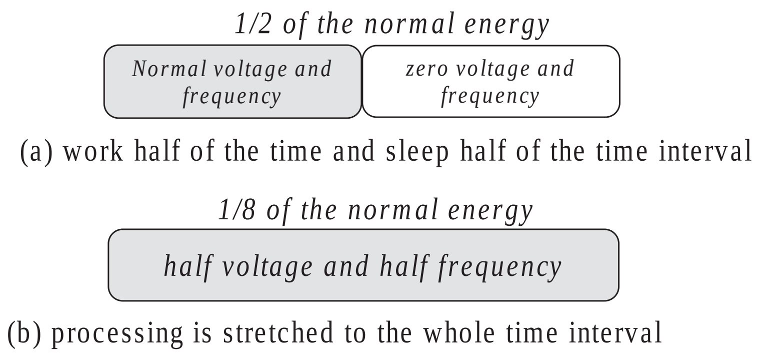

1/2 of the normal energy
Normal voltage and frequency
zero voltage and frequency
(a) work half of the time and sleep half of the time interval
1/8 of the normal energy
half voltage and half frequency
(b) processing is stretched to the whole time interval

Figure 1.4: Benefits of DVFS on energy savings.

[6] aiming to reduce energy consumption of full-duplex Ethernet networks. The proposed ALR solution only works for Ethernet and more research efforts are still needed to extend ALR to be applied for other networks interface types. In addition, switching the interface rate with finer granularity steps (instead of only three steps suggested in [6]) could further enhance the energy consumption efficiency. In this case, two issues need to be addressed for the enhancement of ALR.

First, it can work properly only under the condition that the interfaces of the two end nodes of a link operating with the same data rate. Certain mechanism or protocol is required to negotiate the interface rate between these two nodes. The process of negotiation and switching of different interface rates are expected to be quick so as not to introduce additional packet loss and delay. A two-way handshake procedure could be implemented: receiver (RX) or transmitter (TX) sends a request message of rate change which contains the information of desired rate; upon the receipt of the request, the other node responses with a message with ACK to agree on the change action or NACK to decline the request.

Second, a policy is required to determine when to change the link rate and what is the target rate. The operational states, as well as the interface rate of each state, can be predetermined. The queue length of RX/TX can be the indicator to trigger the transition of the states: the increase of queue length leads to a transition to a state with faster rate, whereas the decrease of queue length results in a lower rate. The system is first operated with the idle state before switching to another rate to prevent potential problems caused by glitches [13].

1.4.3 A Multi-Frequency Scaling Prototype

In this section, we describe a Multi-Frequency Scaling (MFS) scheme for energy conservation of network devices like routers and switches [16]. The frequency of components in a network device is scaled dynamically according to the real time workload. Based on this scheme, we first present a prototype of this scheme in the data path of a general IPv4 router on a real hardware platform - NetFPGA. Experimental results show excellent energy savings at the cost of a tolerable latency under various ranges of traffic loads and indicate the feasibility and possibility of deploying this mechanism into real network devices for energy saving.

In essence, there are two performance metrics for a rate adaptive scheme of network devices, and we present, analyze and evaluate the MFS scheme according to them.

- *Energy saving:* The amount of energy saving is linear to the reduction in rate [16]. Therefore, we use the *rate reduction* to measure the efficiency of energy saving. In particular, we use the average rate reduction which is more meaningful than instantaneous energy saving.
- *Packet delay:* Generally, more time to work on lower frequency, longer delay suffers. Moreover, variation of packet delay brought by frequency switching may have impact on routing protocols that update route using packet delay. Therefore, the rate adaptive scheme must incur limited increase in packet delay. The packet delay consists of *queuing delay* which is used as the metric for delay in this chapter, and a constant amount of time for forwarding and switching.

Basically, a rate adaptive scheme needs to strike a balance between energy saving and packet delay even with highly dynamic traffic loads. In the following sections, we investigate, both in theory and in data result, the proposed MFS scheme with the above metrics. We show that it is feasible to achieve excellent energy saving with tolerable cost in traffic delay using multi-frequency.

Generally, a network device works at two states: *active* state and *idle* state. When the device is actively processing traffic, it works at the active state. When there is no traffic for processing but the device is still powered on, it works at the idle state. Thus, the energy consumption of a general network device could be modeled as:

$$E = P_a T_a + P_i T i, \tag{1.17}$$

where T_a and T_i denote the time spent in active state and idle state respectively, P_a and P_i represent the power consumption in each mode. For both active and idle states, there is a static portion of power draw which is independent from the device's operating frequency, while the rest of power draw indeed depends on the operating frequency:

$$\begin{aligned} P_a(r) &= C + f(r), \\ P_i(r) &= C + \beta f(r), \end{aligned} \tag{1.18}$$

where $f(r)$ reflects the dynamic portion of energy consumption working at frequency r, and C denotes the static portion of energy consumption. The parameter β represents the relative magnitudes of routine work incurred even in the absence of packets to the work incurred when actively processing packets [17]. In general, the dynamic portion of energy consumption depends on the operating frequency and voltage of the network device:

$$f(r) \propto r V_{dd}^2, \tag{1.19}$$

where V_{dd} is the voltage of the device [19]. Hence, energy consumption could be reduced for both states by scaling the device's operating frequency according to its workload.

It is observed that buffers are commonly adopted in current design of network devices as queue for processing jobs. Figure 1.5 shows an example of this, the Framer and Packet Processor in the line card are connected by a buffer. Intuitively, the actual buffer occupancy is considered as indicator for real-time traffic load. Our MFS mechanism uses this indicator for frequency switch. We assume that the hardware supports working at several different rates. The operating rate of devices is dynamically switched according to the buffer occupancy. A range of thresholds are set to evaluate the relative workload of the devices.

To avoid rate oscillation of single threshold, Dual-Threshold scheme is proposed, which

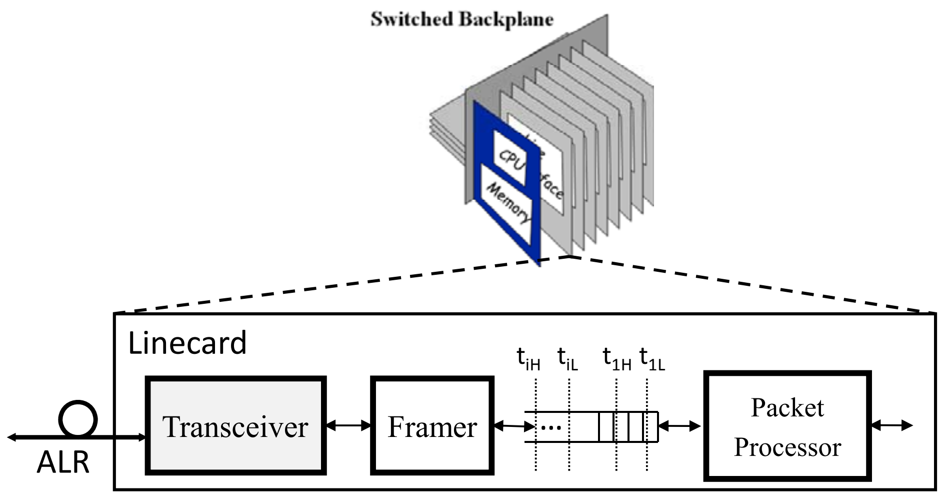


Figure 1.5: Example of components connected by buffers in the line-card.

uses both high threshold t_H and low threshold t_L for rate switch, as illustrated in Fig. 1.6 (a). When component works at low rate mode, the buffer occupancy (denoted as Q_Length in Fig. 1.6) would rise with the increase of traffic load. The transition from low rate to high rate is induced when the buffer occupancy exceeds t_H. On the other hand, when component operate at high speed mode, the queue length would drop while the traffic load decreases. Similarly, a transition from high rate to low rate is caused when the queue length drops below t_L.

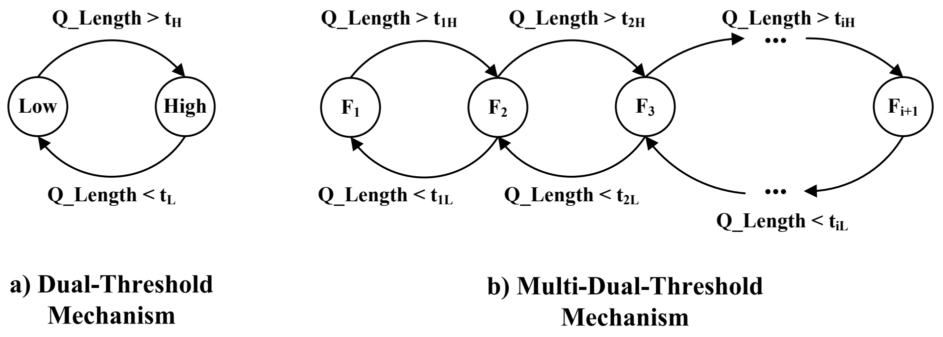


Figure 1.6: State machine for buffer occupancy based frequency scaling mechanism.

Now, we give a detailed description of the implementation of MFS on NetFPGA.

The NetFPGA is a line-rate, flexible and open platform for gigabit-rate network switching and routing, which enables students and researches to build high-performance networking systems using Field-Programmable Gate Array (FPGA) hardware. The core data processing functions are implemented in a modular style, enabling researches to design and build their own functional components independently without modifying original codes [15]. Here, we use NetFPGA-1G card for hardware prototype, which contains one Xilinx VirtexII-Pro 50 FPGA, The core clock of NetFPGA-1G card could be configured to operate at either 125 MHz or 62.5 MHz. Two SRAMs work synchronously with this core FPGA. In our implementation, the core clock rate is set as 125 MHz, which is the default setting of NetFPGA-1G card.

The *Rate Adapter* (*i.e.*, *CLK Adapter* in Fig. 1.7) is the main component for rate adaptation. The CLK Adapter reads the FPGA's core clock as input, and generates a range of exponentially distributed frequencies based on the input, by simply using a clock cycle counter. The buffer is used for frequency switch decision. Frequency transition is achieved by switching to another frequency according to the input clock, and set it as the output. In our implementation in NetFPGA, the frequency switch does not involve much overhead, which suggests frequency switch has negligible impact on packet delay.

The output is used for driving the modules connected to the output of buffer, as shown in Fig. 1.7. To avoid misalignment between input and output clock frequencies, D flip-flop is adopted for buffering the generated output clock. Thus, the actual frequency transition happens one clock cycle after the decision of transition is made.

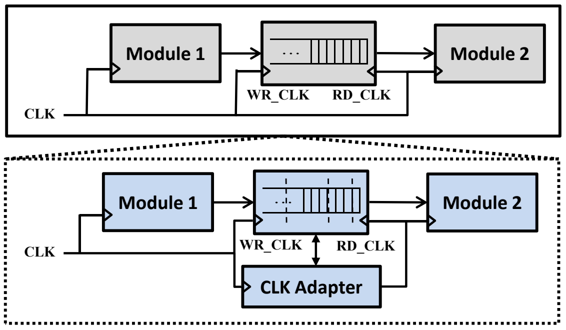


Figure 1.7: Adaptive frequency scaling module.

In the implementation, the CLK Adapters are embedded between *Input Arbiter* and *Output Port Lookup*, and into some sub-modules of *Output Queues*. Six frequencies distribute from 3.096 MHz to 125 MHz in an exponential order of two. The core processing rates are accordingly from 250 Mbps to 8 Gbps. Since not all components in NetFPGA support operating at a frequency different to the default value, it is difficult to directly measure the reduction in energy saving of components supporting frequency scaling. Energy conservation could only be estimated using equation (1.19) in Section 3. However, the rate reduction could be measured by exploiting the *Register System* of NetFPGA. The register interfaces allow software running on host system to send data to and receive data from the hardware modules. A few registers are used for recording operating time of each frequency in the prototype, so that we could estimate average rate reduction.

The power saving percentages of dynamic portion of energy consumption of modules supporting MFS in our prototype are estimated by using (1.19), and are presented in Fig. 1.8. The reason that the minimum saving percentage is around 50% is also because that the full processing rate is 8 Gbps. Since frequency only has influence on the dynamic power consumption, the total energy saving may be less than the results presented in Fig. 1.8. However, the dynamic power consumption is much higher than the static power consumption generally, when there is no rate adaptation in hardware. Thus, MFS would achieve great energy saving in modules supporting it, especially when the average utilization is low. Based on the results presented above, we draw the conclusion that the MFS mechanism is feasible and effective for energy conservation in real systems.

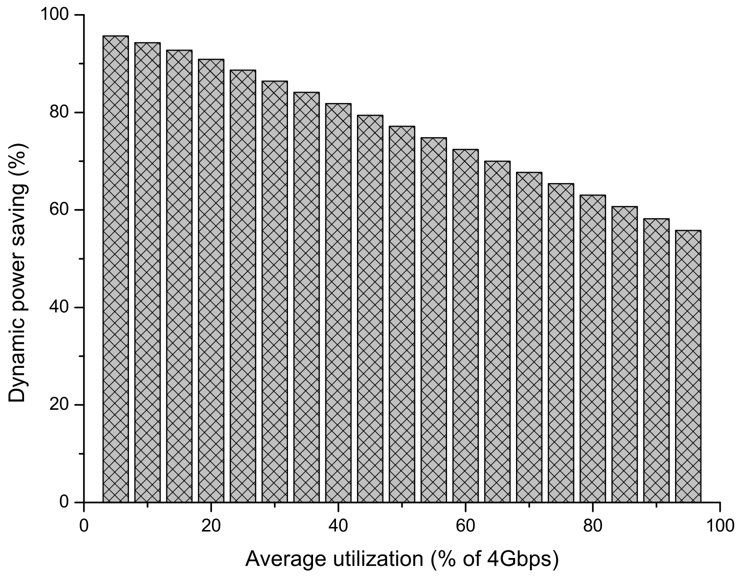


Figure 1.8: Dynamic power savings of modules supporting MFS in MFS Router.

1.5 Power Efficient Architectures for Router Functions

1.5.1 Routing Lookup

In general, two major categories of routing lookup algorithms are used based on Classless Inter-Domain Routing (CIDR) IP addresses: Ternary Content Addressable Memory (TCAM) based schemes and trie-based algorithms. TCAM is a fully associative memory and is capable to return the matching result within a single memory access. Therefore, it dominates today's high-end routers. Although the lookup speed of trie-based mechanisms using Dynamic Random Access Memory (DRAM) or Static Random Access Memory (SRAM) is slower than TCAM-based method, its cost and power consumptions are significantly lower. Estimation shows that the power consumption per bit of TCAM is about 150 times more than that of SRAM. Therefore, it would be desirable to reduce the energy consumption of TCAM-based mechanisms whilst keeping its high performance.

The power consumption of TCAM is high since it is a fully parallel device. When the search key is input, all entries are triggered to perform the matching operations. Current high-density TCAM devices consume as much as 12-15 Watts per chip when all the entries are enabled for search. However, the power consumption of TCAM can be reduced. We observe that not all the entries need to be triggered simultaneously during a lookup operation. The entries that really need to be searched are the ones matching the input key. Thus, one way to make TCAM-based method power efficient is to prevent the non-essential entries from being triggered during a lookup operation. There are two ways to decrease the number of entries triggered during a lookup operation. The first one is to store the entries in multiple small chips instead of a single large one. The second way is to partition one TCAM into small blocks and trigger only one of them during a lookup operation. Exploring these two energy-saving possibilities, multiple small TCAMs instead of only one large TCAM is utilized and the entries inside each TCAM are divided into several blocks. In this way, only one block on one TCAM is activated for routing lookup and so the energy can be saved. The division of routing table so as to fit into the TCAM blocks is the key of this problem, which should evenly allocate the route entries among the TCAM chip, balance the lookup traffic load among the TCAMs and avoid possible prefix conflicts. A detailed mechanism can be found in our previous work[21].

1.5.2 Packet Classifier

Packet classification involves matching several fields extracted from a packet header to a set of pre-defined rules in order to determine the follow-up action to be performed on the packet by networking devices. As one of the key functions of a router/switch, packet classification is widely used in applications such as policy-based routing, QoS, Virtual Private Networks (VPN), network security (e.g., firewall, and intrusion detection), and sophisticated traffic billing. Transmission line speeds keep increasing. For example, the state-of-art line rate of OC-768 (40 Gbps), used in optical fiber backbone networks, demands a worst case processing speed of 125 million packets per second. In addition to meeting the ever-increasing line rates, growing attention has been paid to reducing the power consumption in the design of next generation networking devices, which make it desirable for the packet classifier to consume as little energy as possible.

The most commonly used header fields in packet classification form a 5-tuple which includes source IP address and destination IP address for prefix matching, source port and destination port for range matching, and protocol number for exact or wildcard matching. The typical size of a rule-set ranges from hundreds to millions of rules, depending on the application and location of router/switch. Due to the complexity of packet classification, TCAM is frequently used, which can guarantee millions of searches per second by examining all rules simultaneously. However, the high power consumption caused by the parallel comparison in TCAM makes it less likely to be incorporated in the design of future power-efficient networking equipment.

We aim to design a packet classifier that meets the challenges of both high speed and low power consumption [13]. In order to achieve this goal, we choose to implement a decision tree based packet classification algorithm (HyperCuts) in hardware, so that TCAM is replaced with energy efficient memory devices such as SRAM or DRAM. This type of algorithm recursively cuts the hyperspace represented by the rule-set into smaller hyper-boxes called Regions along some selected dimensions, which form a decision tree, until the number of rules contained in the resulting hyper-boxes is smaller than a predefined threshold. When a packet is received, the classifier traverses down the decision tree, based on the value of header fields, until a leaf node is found and searched for the matching rules. The structure and parameters of the packet classifier are carefully selected to make sure that only a small number of memory accesses are needed in order to guarantee the throughput.

The major parts of the packet classifier consist of a Tree Traverser and Leaf Searcher as shown in Fig. 1.9. Each internal node contains the cutting scheme to be performed on the hyperspace represented by this node, and the starting address of each child node. Other information, such as whether the child node is a non-empty leaf, is also coded. In each leaf node, rules are stored in the order of priority, along with their IDs. When the cutting scheme is fetched and interpreted, the Region corresponding to the header fields is selected, i.e. the starting address of this child node will be read out from memory and used for the next round of tree traversing. If a non-empty leaf is encountered, the control is passed to Leaf Searcher. The rules are compared against packet header fields one by one, until either a matching rule is found or all of the rules have been examined.

Power consumption of packet classifier can be further reduced by exploiting the fluctuation in network traffic. It has been observed that high level of variation is common in network traffic. For example, traffic volume during the night might be less than one third of the peak day rate. This phenomenon implies that packet classifier does not have to always run at its highest possible clock speed. When the packet arrival rate is low, the packet classifier can reduce its working frequency to an appropriate level in order to save energy. In our project, we use a header buffer to monitor the change in network traffic, in which the number of outstanding packet headers waiting for classification works as an indicator of the frequency required for the packet classifier.

Our packet classifier can be easily implemented in hardware (FPGA or ASIC), working as an individual component on a line card or integrated into a network processor. The experiments performed on sample implementations in Altera FPGA and ASIC show that

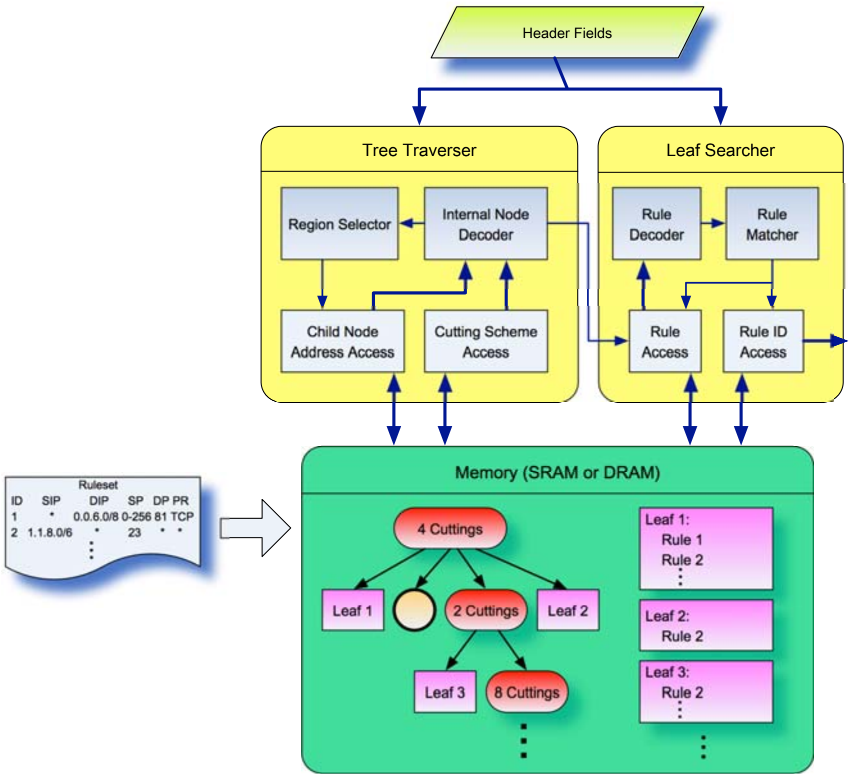


Figure 1.9: Reduced power vs. the number of queues.

our packet classifier can achieve packet classification throughput of up to OC-768 line rate with significant less power consumption than TCAM implementations.

1.5.3 Packet Queuing

The queuing process in the buffer is introduced in the routers to temporarily store the packets which cannot be processed immediately due to network congestion. Per-flow queuing mechanism is suggested to achieve advanced (QoS) guarantees by segregating individual flows into dedicated queues. Commercial products adopt "brute-force" implementation of per-flow queuing by deploying a separate physical queue for each in-progress flow.

The authors in [14] observe that the number of active flows in a packet scheduler is measured typically in hundreds even though there may be tens of thousands of flows in progress. With such a recognition, Dynamic Queue Sharing (DQS) mechanism, which provides dedicated queues for only simultaneous active flows instead of all the in-progress flows, is proposed in [9, 11]. DQS is demonstrated effective to reducing the required physical queues from millions to hundreds, and hence to reduce the energy consumption.

If packets of a flow are queued in the router buffer, it is considered as an active flow and a dedicated physical queue is assigned to it. In order to keep all the arrival packets of a certain flow to be in one physical queue, an Active Flow Mapping (AFM) table is introduced to record the mapping information between active flows and physical queues. When a packet of a flow arrives, DQS checks if the flow is in the AFM table. If yes, the packet is pushed into the corresponding queue; otherwise, a queue allocator creates a new queue to buffer the packets from this flow and correspondingly a new mapping entry between the flow and the queue is added to the AFM table. Once a physical queue becomes empty, the queue and its mapping to the flow will be withdrawn immediately. Instead of maintaining all the states of the flows, DQS only needs to manage a small number of physical queues and an AFM table. To achieve more efficient operations on the AFM table, the AFM table is firstly divided into a number of small AFM sub-tables by using hashing operations and each sub-table is organized through linked-list or binary sorting tree.

Since less queues are required, the peak power of packet queuing is reduced. We implement prototypes of DQS and brute-force per-flow packet queuing in Field Programable Gate Array (FPGA)[2] in order to check the power consumption. With the help of Stratix PowerPlay Early Power Estimator, we are able to evaluate the power consumption of the prototypes. Considering the power consumption of the internal logic and on-chip memory to support 8,000 flows, the estimator's results indicate that the power consumption with DQS is about 23.3% less than the brute-force approach. If one aims to support more flows, more gain on energy-saving can be achieved through DQS. Fig. 1.10 depicts how much power can be reduced by employing DQS compared with brute-force method.

[2]We use ALTERA Stratix EP1S80F1508C5 for implementation.

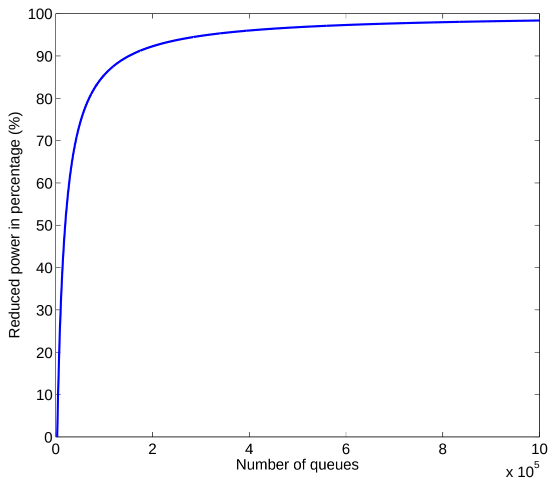


Figure 1.10: Reduced power vs. the number of queues.

1.5.4 Traffic Manager

Within a line-card of a high-end router/switch, the traffic manager (TM) stores the arrival packets in its off-chip memory (e.g., a DDR-3 or RLDDR-2 memory). Even though only a small portion of the memory is used for queuing at most of the time, TM cannot turn off the remaining portion of the memory to reduce power consumption.

Previous research has reported that the buffer size in Internet core router can be very small [3] and the buffer size can be reduced to 20-50 packets if we are willing to sacrifice a fraction of link capacity, and if there is a large ratio between the speed of core and access links. Therefore the TM can utilize a small-size on-chip memory to buffer the packets most of the time, and activate the off-chip memory to store the packets only when the on-chip memory is about to overflow. In such a design, the off-chip memory and its corresponding memory controller are turned off by clock gating when there is no or light contention. With this idea in mind, we design and implement a *dynamic on-chip and off-chip scheduling scheme*, named *DPM*, to reduce both the peak and average dynamic power consumptions of the TM function.

In general, the packets need to be queued in both ingress and egress. For illustration, we first describe the architecture of the TM in the ingress direction as shown in Fig. 1.11.

There are four key modules: segmentation, per-flow manager, packet manager and scheduler. Given the switch fabrics can be optimized for switching fixed-sized data units (cells), and the packets arriving at the traffic manager are usually variable-length packets, segmentation module will segment the packets into fixed length cells for further processing[3]. In order to provide advanced QoS, the arrival packets will be queued in a per-flow manner and the packets from a same flow are queued in one dedicated queue. Per-flow manager takes charge of packets to be written into the right queues, i.e., to determine in which queue an incoming packet will be kept in the buffer. The scheduler is responsible for reading out the packets from a certain queue in the buffer.

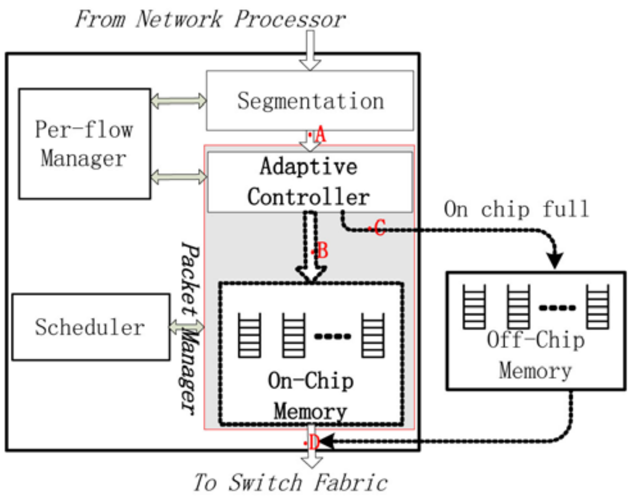


Figure 1.11: A schematic system of TM (Ingress direction)

The TM's working flow-chart is described as follows. A packet from the upstream module[4] of TM is first segmented into fix-sized data units. Meanwhile, the flow ID of the packet is sent to the per-flow manager, where the address of the queue to store the packet is determined. And then the packets along with the address of the queue are sent to DPM (Point A in Fig. 1), where the packet management is conducted. DPM operates two kinds of packet queuing: on-chip memory queuing and off-chip memory queuing. If the adaptive controller (AC) in DPM detects that there is enough room in on-chip memory to hold the incoming packets, the packets are stored in the on-chip memory (Point B) directly. Otherwise, the packets will be kept in the off-chip memory (Point C). The detailed schemes to determine the switching between the on-chip memory and the off-chip memory will be described in later.

[3]The packets can also be segmented before its arrival to TM. In this case, the traffic is fixed-length cells.
[4]It is usually a network processor in commercial products.

In the on-chip memory queuing, it adopts the aforementioned DQS [9] technique to let all the packets from the same flow to be stored in a same physical queue so as to improve QoS, as mentioned above. In the off-chip memory queuing, we can either adopt DQS technique to support per-flow queuing or utilize the class information of the flow to implement class-based queuing management. Scheduler determines which flow queue will be sent out from its head of line packet to the *Switch Fabric* (Point D).

The switching mechanism between the on-chip memory and off-chip memory is the key part of the system design and is described as follows.

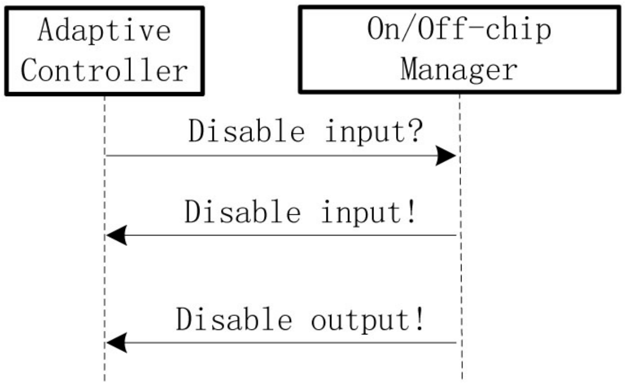


Figure 1.12: Scheduling between adaptive controller and on/off chip manager

- **Switching from on-chip to off-chip** The adaptive controller described in Fig. 1.11 always queries the free memory size in the on-chip memory. As illustrated in Fig. 1.12, if the free size is smaller than T_{on}(T_{on} is a threshold for on-chip free memory size), the controller will send a "disable input?" message to the on/off-chip manager to attempt to disable the input direction of the on-chip memory. When the manager gets the message, it will not disable its input direction of on-chip memory until it has received a whole packet. And then the manager sends a "disable input!" message to controller so as to enable the input direction of the off-chip memory. When all the packets in the on-chip memory are scheduled out, the manager sends a "disable output!" message to controller and the on-chip memory is disabled.
- **Switching from off-chip to on-chip** The adaptive controller keeps querying the un-free memory size in the off-chip memory. If the un-free size is smaller than T_{off}, the DPM starts the switching operation. Similar to switching from on-chip to off-chip,

the on/off chip manager will first disable the input of the off-chip memory and enable the input of the on-chip, and then enable the output of the on-chip memory when the off-chip memory is empty.

If the on-chip memory size is M_{on}, and the size of the packets buffered in the packet manager oscillates around $M_{on} - T_{on}$, DPM will switch frequently between the on-chip memory and the off-chip memory. To prevent the frequent switching, the threshold T_{off} for the off-chip memory should be smaller than $M_{on} - T_{on}$.

In this section, we have presented an energy-efficient packet management architecture named DPM. DPM can be utilized to construct the TM chip in a router with less peak and average dynamic power consumption while providing improved delay performance. Compared with the packet management schemes in traditional commercial TM chips, real prototype testing shows DPM has several advantages: first, the dynamic power of DPM is reduced by 60% as demonstrated through prototype testing with a small-sized on-chip memory; second, the peak power of DPM is reduced by 27.9%; third, DPM facilitates a reduced average delay, meanwhile the required total memory and the registers are both reduced accordingly. Please refer to [4] fort the details about the experiments and evaluations.

1.6 Summary

A great deal of researches have contributed to the energy efficiency on battery-operated devices in the area of wireless communications, but until recently, the energy consumptions for underlying network infrastructure start to get more attentions since we begin to be aware of the significant fraction of energy consumed by the Internet over the entire energy we consumed.

Router is the main equipment that is used to construct the Internet. We argue that the architecture of the routers to construct the Internet can be improved so as to fully utilize the chances to save the energy related to transmission equipments. By exploring the means to control the energy consumptions of routers, the Internet could become energy efficient.

GRecRouter presented in this chapter seeks opportunities and means to notably reduce the power dissipation at network level, node level and function level.

References

[1] Raffaele Bolla, Roberto Bruschi, Franco Davoli, and Andrea Ranieri. Energy-aware performance optimization for next-generation green network equipment. In *PRESTO '09: Proceedings of the 2nd ACM SIGCOMM workshop on Programmable routers for extensible services of tomorrow*, pages 49–54, New York, NY, USA, 2009. ACM.

[2] Davidsarokin. Energy use of internet, 2007.

[3] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Routers with very small buffers. In *INFOCOM 2006*, pages 1 –11, april 2006.

[4] Jindou Fan, Chengchen Hu, Keqiang He, Junchen Jiang, and Bin Liu. Reducing power of traffic manager in routers via dynamic on/off-chip scheduling. In *INFOCOM 2012*, Berkeley, CA, USA, Mar. 2012.

[5] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S.C. Diot. Packet-level traffic measurements from the sprint ip backbone. *Network, IEEE*, 17(6):6 – 16, nov.-dec. 2003.

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

[7] M. Gupta and S. Singh. Using low-power modes for energy conservation in ethernet lans. In *INFOCOM 2007*, pages 2451 –2455, may 2007.

[8] Maruti Gupta and Suresh Singh. Greening of the Internet. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 19–26, New York, NY, USA, 2003. ACM.

[9] Chengchen Hu, Yi Tang, Xuefei Chen, and Bin Liu. Per-flow queueing by dynamic queue sharing. In *IEEE INFOCOM 2007*, pages 1613 –1621, may 2007.

[10] Chengchen Hu, Chunming Wu, Wei Xiong, Binqiang Wang, Jiangxing Wu, and Ming Jiang. On the design of green reconfigurable router toward energy efficient internet. *Communications Magazine, IEEE*, 49(6):83 –87, June 2011.

[11] Chengchen Hu, Tang Yi, Kai Chen, and Bin Liu. Dynamic queuing sharing mechanism for per-flow qos control. *IET Communications*, 4(4):472–483, 2010.

[12] IBM. Ibm ilog cplex optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[13] Alan Kennedy, Xiaojun Wang, Zhen Liu, and Bin Liu. Low power architecture for high speed packet classification. In *ANCS '08: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pages 131–140, New York, NY, USA, 2008. ACM.

[14] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. In *SIGMETRICS '05: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 217–228, New York, NY, USA, 2005. ACM.

[15] J.W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and Jianying Luo. Netfpga–an open platform for gigabit-rate network switching and routing. In *IEEE International Conference on Microelectronic Systems Education, 2007.*

[16] Wei Meng, Chengchen Hu Yi Wang, Keqiang He, and Bin Liu. Greening the internet using multi-frequency scaling schemes. In *IEEE AINA 2012*, Fukuoka, Japan, Mar. 2012.

[17] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *Proc. of NSDI, 2008.*, pages 323–336, Berkeley, CA, USA. USENIX Association.

[18] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 323–336, Berkeley, CA, USA, 2008. USENIX Association.

[19] Bo Zhai, David Blaauw, Dennis Sylvester, and Krisztian Flautner. Theoretical and practical limits of dynamic voltage scaling. In *In DAC 04: Proceedings of the 41st annual conference on Design automation*, 2004.

[20] Mingui Zhang, Cheng Yi, Bin Liu, and Beichuan Zhang. Greente: Power-aware traffic engineering. In *18th IEEE International Conference on Network Protocols (ICNP 2010),*

pages 21 –30, Oct. 2010.

[21] Kai Zheng, Chengchen Hu, Hongbin Lu, and Bin Liu. A tcam-based distributed parallel ip lookup scheme and performance analysis. *IEEE/ACM Trans. Netw.*, 14(4):863–875, 2006.