Reducing Power of Traffic Manager in Routers via Dynamic On/Off-chip Scheduling

Jindou Fan, Chengchen Hu, Keqiang He, Junchen Jiang, Bin Liu†
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China

***Abstract*—Green networking in the Internet becomes increasingly important. In a high-performance router, the dominant power consumer on the Internet, half of its total power usage goes into the line-cards, where the traffic managers inside consume most of it. In this paper, we propose an energy-efficient design on the traffic manager architecture for packet buffering and storage. Unlike traditional routers where packets are always kept in off-chip memory, we propose a dynamic on-chip and off-chip scheduling mechanism, called Dynamic Packet Manager (DPM), to reduce both peak and average power consumption caused by the traffic manager. DPM buffers packets in a small on-chip memory in the light-traffic period, and activates the off-chip memory on when the on-chip memory is to overflow. In this design, when the traffic is light, the off-chip memory is put into power saving state by clock gating so that the average power consumption is reduced. With an on-chip flow based and off-chip class-based design, DPM can save one off-chip memory otherwise used for the per-flow index information storage, therefore further reduce the peak power usage. We present the theoretic analysis guiding the implementation of the DPM mechanism. Experiments on three prototypes implemented on different hardware show that the peak and average power consumptions can be reduced by 27.9% and 37.5% respectively, along with less on-chip memory cost. Besides, the traffic manger with DPM shows better performance on average packet scheduling delay than the one without DPM.**

I. Introduction

Energy efficiency of the Internet gains increasing attention. According to the research by Uclue.com [1], the Internet contributed to 9.4% (5.3%) of the total energy consumption in the U.S. (globally) at 2007. Routers and switches, the infrastructure components of the Internet, contribute a great deal to its energy consumption. For example, a high-end router like Cisco CRS-1 may consume 1.02 MWatts annually [2]. In a modern backbone router, the line-cards and the cooling system dominate the equipment's power consumption [3]. Within a line-card, the traffic manager (TM), the network processor (NP), and their connecting components are the main energy consumers. In this paper, we explore the potential energy savings on TM chips.

In current design practice, TM stores incoming packets in its off-chip memory (e.g., DDR-3 or RLDDR-2 memory). Even though the memory has a low utilization for most of the time, TM cannot turn off the unused portion to reduce power usage.

We observe that a router needs large memory only when network congestion happens locally; when there is no congestion, a small buffer is enough since packets depart quickly in the routing queue. Measurement results showed that the maximal average link utilization of the Internet backbone network is 40% or less [4]. Intuitively, the low link utilization means that traffic arrival intensity at the ingress is low, and the probability of packet blocking in the inlet/outlet of TM chips is also small. Therefore, with a small on-chip memory, TM can deactivate the large off-chip memory in most of the time, and activate the off-chip memory only when the on-chip memory is about to be overflowed. Small-sized on-chip memory consumes much less power than large off-chip memory. For example, the power consumption of a 144Kb on-chip memory in FPGA is only 3.2mW, while that of a 2GB off-chip DDR3 memory is around 1300mW. With those observations in mind, we design and implement a dynamic on-chip and off-chip scheduling scheme, called *Dynamic Packet Manager* (DPM), to reduce both peak and average power consumption of the TM component. In such a design, the off-chip memory and its corresponding memory controller are turned off when there is no or light contention.

As a traditional TM needs an extra off-chip memory to store queue information for per-flow queueing, and DPM can support per-class queuing in the off-chip memory and per-flow queuing in the on-chip memory, which can lead to save the number of off-chip memories. As a result, DPM can reduce the peak power consumption accordingly. And it is worthy noting that reducing the peak consumption is essential to maintain supply voltage levels, increase reliability, reduce the size of heat sinks and the cooling fans capacity budget as well as minimizing the systems packaging cost.

Although the idea of using on-chip memory for power saving has been explored in computing systems, we are the first to apply it into router energy efficiency, to our best knowledge. This idea is further combined with the exploitation of Internet traffic dynamic patterns to achieve remarkable power saving on routers. Besides, we address the following technical questions in the design of DPM: **Q1)** when and how is the packet queueing switched between the on-chip and off-chip memory? **Q2)** how much on-chip memory should be provisioned for traffic smoothing? **Q3)** how much peak and average power usage saving can be realized for a TM with DPM? **Q4)** What is the extra cost in facilitating such a power

† Corresponding author: liub@tsinghua.edu.cn.
Others: fanjindou@gmail.com, huc@ieee.org, hekeqiang@gmail.com, junchenj@cs.cmu.edu (now a PhD student in CMU).

This work is supported by NSFC (61073171, 60873250, 60903182), Tsinghua University Initiative Scientific Research Program, the Specialized Research Fund for the Doctoral Program of Higher Education of China(20100002110051).

saving mechanism? To answer those questions, we provide both theoretical analysis and prototype evaluation on a real hardware system.

Specifically, this paper makes the following contributions:

1) We propose a memory switching mechanism for DPM. With such a scheme, we can reduce both the peak and dynamic power consumptions of the TM in a router;
2) Theoretical analysis about the relationship between energy saving and the size of on-chip memory is derived to guide the design of the TM prototype;
3) We implement the DPM logic in an FPGA chip and develop a corresponding line-card prototype. Based on the prototype, we demonstrate the efficiency of DPM: about 27.9% peak power and 37.5% average dynamic power reduction can be achieved compared with the existing TM scheme, in addition to the reduced on-chip memory expense.

The rest of the paper is organized as follows. After surveying the related work in Section II, we describe a system design of adaptive on-chip/off-chip packet management in Section III. The modeling analysis is discussed in Section IV. The detailed system simulation and experimental evaluation are conducted in Section V. Finally, we draw the conclusion in Section VI.

II. Related Work

In the context of greening the Internet, the literatures can be roughly classified into two categories: 1) reducing the energy consumption of the Internet through end-to-end control and, 2) reducing the power of the network nodes.

A. End-to-end Solutions

The basic ideas of saving energy over Internet through end-to-end solutions are discussed in [5]. Over-provisioning of the link bandwidth and massive link redundancies are common in network planning to harness burst traffic and to survive the network in failures. The idea 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. A recent concrete study in this direction can be found in [6].

B. Power Reduction on Network Nodes

For a traditional network intermediate node, e.g., a router, it has several major functional units: interface, route lookup and classification engine, packet processor, switch fabric, and etc. Related work seeks power-efficient solutions in the system level and the functional unit levels.

In the system level, the authors in [7] investigate the sleeping mechanisms for Ethernet devices (e.g., LAN switches/Ethernet NIC), which attempts to identify the inactive periods of the links (when the traffic rate is low) and put the associated devices into sleep in such periods.

In [8], Dynamic frequency and voltage scaling technology introduced to reduce the dynamic power of high speed packet classifier inside a router. A power efficient architecture dynamically adjusts the clock frequency of the classifier to match fluctuations in traffic on a router's line-card.

Dynamic queue sharing (DQS) mechanism is proposed in [9], and its implementation on FPGA demonstrates its power efficiency [10]. DQS provides dedicated physical queues only for the active flows instead of all the in-progress flows. Considering the power consumption of the internal logic and on-chip memory, the estimator's results indicate that the power consumption with DQS is reduced by about 23.3%, much less than the brute-force approach.

The switch fabric is a very hot spot of the router, and a good understanding on its power consumption will benefit the design of a low power router. With this purpose, a framework to estimate the power consumption on switch fabrics is studied in [11]. Four switch fabric architectures are analyzed under different traffic throughput and different numbers of ports.

All the existing industrial TM products store packets in off-chip memories, and frequency scaling technology has limited effects on energy saving since the off-chip memory has to work on a frequency larger than a required-minimal threshold, such as the most RLDDR memories do in the market.

To the best of our knowledge, we are the first to propose using the technology of dynamic on-chip and off-chip scheduling to reduce both the peak and average dynamic power consumptions of a TM chip.

III. System Design

In this section, we describe the system design and answer the technical question **Q1** mentioned in Section I.

A. The Dynamic Packet Manager related System Structure

The key function of the TM chip is to store the packets during the transit contention periods. In general, the packets need to be queued in both ingress and egress direction. For illustration, we first describe the architecture of the TM in the ingress direction as shown in Fig. 1. There are four key modules: segmentation, per-flow manager, packet manager and scheduler. Given that 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 segments the packets into fixed-length cells for further processing[1]. In order to provide advanced Quality of Service (QoS), the arrival packets will be queued in a per-flow manner and the packets from a same flow are queued in a dedicated queue individually. 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.

The TM's working flow-chart is described as follows. A packet from the upstream module[2] of TM is first segmented into fix-sized data units. Meanwhile, the flow ID of the

[1]The packets can also be segmented before its arrival to TM. In this case, the traffic is fixed-length cells.

[2]It is usually a network processor in commercial products.

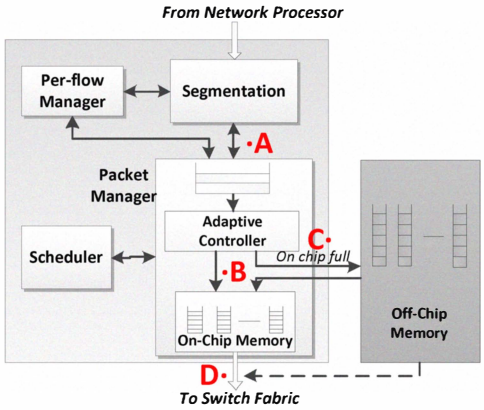


Fig. 1. A schematic system of TM (Ingress direction)

packet is sent to the per-flow manager where the address of the queue to store the packet is determined. And then the packet and the address of its corresponding 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 Section III-C. In the on-chip memory queuing, we adopt the Dynamic Queue Sharing (DQS) [9] technique, proposed in our previous work, to let all the packets from the same flow to be stored in a same physical queue so as to improve QoS. 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).

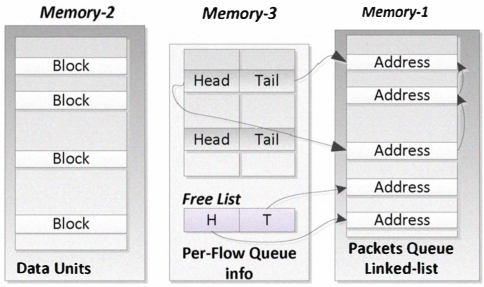
In the ingress direction, if the switch fabric's speedup is greater than one, ideally there should be no packets blocked in the on-chip memory, because in normal case the output speed is faster than the input's except when a serious switching contention occurs or temporary network congestion happens. So we expect that DPM will work in the on-chip mode most of the time. We design an appropriate-sized on-chip memory to amortize the packets accumulation due to this kind of contention, so as to minimize the switching opportunity between the on-chip memory and the off-chip memory.

In the egress direction, the packet management mechanism works similarly as in the ingress direction. Fortunately, as we have mentioned earlier, the average link utilization is low, making the probability of packets being stored in the off-chip memory extremely low.

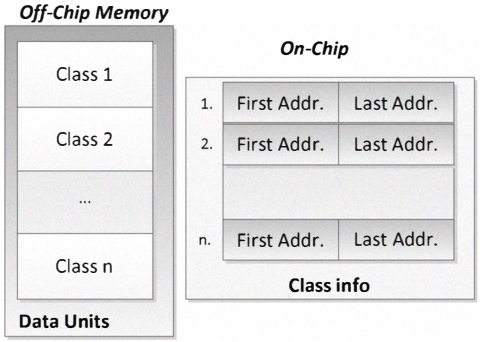
Here, we need to emphasize that it is the congestion degree that decides the switching between on-chip and off-chip memory, rather than the traffic load, although the arrival traffic intensity does have an indirect influence on the packets accumulation in on-chip memory. Intuitively, if the input traffic volume and the output traffic volume to/from the on-chip memory get balanced, i.e. no additional packets accumulation appears, the incoming packets are still stored in the on-chip memory, despite of the heavy arrival load.

B. The Data Storing Structure in Packet Manager

In this subsection, we introduce the data storing structure in packet manager to help understand how and why our DPM can save power.



(a) Packet managing scheme with DQS



(b) Packet managing scheme with class-based queue

Fig. 2. Data storing structure for packet management

Fig. 2 depicts the packet storing structure in TM. Fig. 2(a) shows the packet storing structure with DQS. Memory-2 is split into fixed-size blocks to store the fixed-size data units. To facilitate per-flow queuing, all the data units stored in a same queue, as well as the unused data units, are organized in a linked-list manner, causing an additional linked-list data structure (memory-1, the address of memory-1 is corresponding to the block address in memory-2). Besides, all the flow queues and the unused data units queue are organized by their Head and Tail address index, resulting in an additional flow index table (including the free list table, stored in the memory-3). In summary, three kinds of data structure need to be implemented to achieve advanced per-flow queuing, accordingly three pieces of RAMs.

In traditional TM, memory-2 is an off-chip memory, which is usually a large capacity DRAM. To support per-flow queuing, the linked-list data structure stored in memory-1 also occupies a large memory space. For example, when employing the MT49H16M18C RLDRAM with 288Mb size as memory-2, and the size of the data unit is 64B, we will have 512K

memory blocks in memory-2. As a result, the address width of the block is 19 bits, and the linked-list information will cost 19 bits×512K = 9.5Mb memory space. Therefore, an additional off-chip memory-1 is needed, which is usually QDR_RAM or high speed SSRAM. By DQS, we only need to maintain several hundreds of physical queues to support per-flow queuing. Therefore, the flow index table in memory-3 is very small and can be stored in an on-chip memory.

In the TM with DPM, we set a small-sized on-chip memory to hold the incoming packets. Only when the packets occupation exceeds a pre-defined threshold in the on-chip memory, will the packets be stored in the off-chip memory. So the TM with DPM has two packet managers: on-chip manager and off-chip manager. The first one supports per-flow queuing and buffers packets in the on-chip memory (three kinds of memory shown in Fig. 2(a) are all on-chip memories). The other one has the same structure as the traditional TM. It is worthy noting that, as the access latency of on-chip memory and off-chip memory are different, it's very complicated to maintain the linked-list structure if we make the two types of memory into one packet manager structure. Therefore, in our design, the on-chip manager and off-chip manager are two independent managers, but they will not incur much cost.

Most of the time, on-chip memory has enough room to keep the input packets and supports per-flow queuing. When congestion occurs, the buffer size and the number of flow maintained in the TM can be extremely large. It will cost a lot of memory resources to support per-flow queuing. Therefore, to provide advanced QoS and to save much more power consumption, we can adopt two-stage queue manager in the TM. First, we can divide the traffic into classes, such as DiffServ Services [12] or application-based classification [13], and adopt class-based queuing mechanism rather than per-flow queuing mechanism for the off-chip memory, as shown in Fig. 2(b). And then the packets stored in the off-chip memory can be sent to the on-chip memory to support per-flow queuing manager. In this structure, the off-chip memory is only divided into a few sub-blocks based on the flow class information. Instead of adopting per-flow queue manager mechanism, the packets belonged to the same class are stored in one off-chip memory sub-block in order. Therefore, the DPM only needs to maintain the first and the last used off-chip memory address for each class. Compared with the DQS structure shown in Fig. 2(a), this structure only consumes one off-chip memory and, the on-chip logic is much simple. This structure can reduce the peak power consumption of the TM accordingly.

It is not difficult to imagine that DPM can achieve power saving due to the sparing two physical memories as well as their memory controllers.

Here, we have a design issue, i.e., how to determine the size of the on-chip memory to achieve a perfect power saving. Intuitively, large on-chip memory will keep the packets staying in on-chip memory longer, while incurring more cost; small on-chip memory will be economic but has a large probability that the on-chip memory overflows. Setting an appropriate-sized on-chip memory will be a trade-off between the cost and the power saving gain. This issue will be further studied in Section IV analytically and, experimentally in Section V.

As described above, there are three kinds of queue manager in the TM, named PM-A, PM-B and PM-C. PM-A is the traditional packet manager which always buffers the packets in the off-chip memory. PM-B with DPM supports per-flow queuing in the off-chip memory, PM-C with DPM adopts two-stage queue manager mechanism.

C. Switching between On-chip and Off-chip Memories

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.

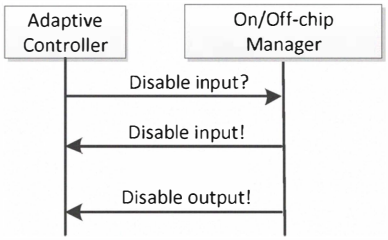


Fig. 3. Scheduling between AC and on/off chip manager

- **Switching from on-chip to off-chip** The AC described in Fig. 1 always queries the free memory size in the on-chip memory. As illustrated in Fig. 3, 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. In PM-B, 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. Differently, in PM-C, the on-chip memory always read packets from the off-chip memory without being disabled.
- **Switching from off-chip to on-chip** As PM-B and PM-C adopt different mechanism in the on-chip memory, their switching methods are described below respectively.
  - **PM-B** The AC keeps querying the occupied memory size in the off-chip memory. If the occupied 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.
  - **PM-C** when the off-chip memory is empty, the AC keeps querying the free memory size in the on-chip memory. If the free size is larger than T_{on2}, the off-chip memory will be disabled.

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 $M_{on} - T_{on}$, and T_{on2} should be larger than T_{on}.

IV. Modeling And Performance Analysis

In this section, we theoretically analyze the proposed DPM and answer the technical question **Q2** in Section I.

A. System and Flow Model

To analyze the performance and exploit the technical questions in Section I from the theoretical aspects, we simplify the TM system (in both the ingress and egress directions) as a queuing model. In our switch model, the packets first wait in the ingress queues, and after going through the switch fabric, they will be buffered in the egress queues.

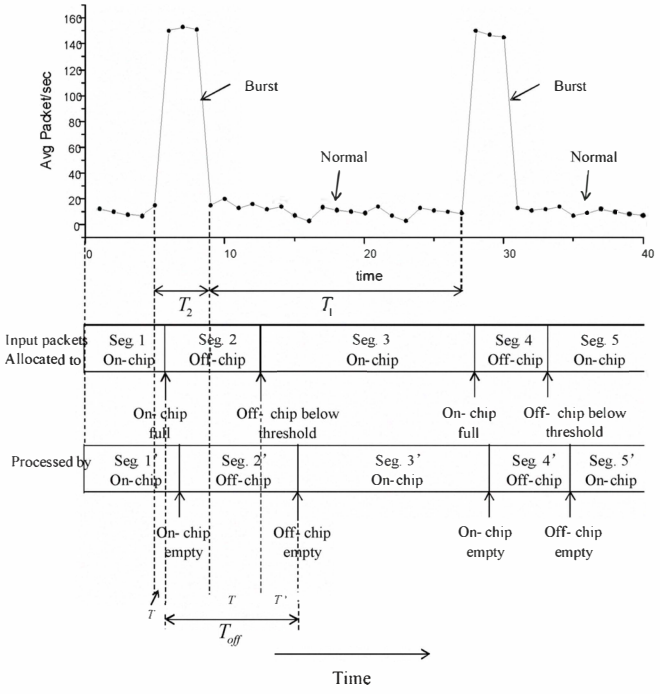


Fig. 4. An illustration of how input packets are allocated to and processed by on-chip and off-chip memories

In the proposed architecture and memory management algorithm, DPM consists of a small-sized on-chip memory and a large off-chip memory. Each time a packet is received, it's stored either in on-chip or off-chip memory. The two memories work as two independent queues, namely on-chip queue and off-chip queue, each processing the input packet by certain scheduling algorithm (e.g., FIFO). The input packets are allocated into two memories in the way illustrated by Fig. 4 (We theoretically analyze PM-B in this subsection, and PM-C has the similar result). At any time, only one queue will work.

We describe the workflow of the two queues from a modeling view as the following. Initially, two queues are both empty and packets are sent to on-chip queue until the queue is full (Seg. 1 in Fig. 4), and then new packets are sent to off-chip queue (Seg. 2). After the on-chip queue consumes all its packets (Seg 1'), off-chip queue starts to process its packets (Seg 2') and new packets continue being sent to the off-chip queue until the number of packets in off-chip queue falls below a threshold. Then new packets are sent to on-chip queue (Seg. 3).

We use Poisson distribution to characterize normal traffic and burst, respectively, with different values of average packets per second, λ_1 for normal segments and λ_2 for bursty segments. And the queues process the packets in FIFO order. A bursty segment occurs every T_1 seconds and lasts for T_2 seconds. We assume that the average number of packets leaving the two queues have the same μ. We also assume that the length of the on-chip queue is L and the off-chip queue has an infinite length.

The design of DPM leverages that most traffic consists of normal segments and bursty segments, with any bursty segment occurs between two normal segments. The on-chip queue shall be sufficient to handle normal segments in order to minimize the usage of off-chip memory, while during a bursty segment the off-chip queue is invoked until the burst ends (when the number of packets in off-chip queue is below a threshold w). Therefore, $\lambda_1 < \mu < \lambda_2$.

B. Analyzing the Rate of Off-chip Memory Usage

We define r_{off} as the ratio of the time during which the off-chip queue works to the whole runtime. The run time of off-chip queue, denoted by T_{off} in Fig. 4, starts at the moment that on-chip queue is full and ends at the moment that off-chip queue is empty. We have

$$r_{off} = \frac{E(T_{off})}{T_1 + T_2}$$

where $E(T_{off})$ is the expectation of T_{off}. We denote the start (end) time of T_i by $T_i^{\triangleleft}(T_i^{\triangleright})$, the start (end) time of T_i' by $T_i'^{\triangleleft}(T_i'^{\triangleright})$, and the start (end) time of T_i'' by $T_i''^{\triangleleft}(T_i''^{\triangleright})$, where $i = 1, 2$. Then

$$T_{off} = T_2 - \Delta T^{\triangleleft} + \Delta T^{\triangleright} + \Delta T'^{\triangleright}$$

where $\Delta T^{\triangleleft} = T_2'^{\triangleleft} - T_2^{\triangleleft}$, $\Delta T^{\triangleright} = T_2'^{\triangleright} - T_2^{\triangleright}$, $\Delta T'^{\triangleright} = T_2''^{\triangleright} - T_2'^{\triangleright}$. So it suffices to have $E(\Delta T^{\triangleleft})$, $E(\Delta T^{\triangleright})$, $E(\Delta T'^{\triangleright})$.

We first compute $E(\Delta T^{\triangleleft})$. $\Delta T^{\triangleleft}$ is the time interval between a burst starts and the on-chip queue is full. Let's denote by p_i the probability that the on-chip queue has i packets during normal segments. By queuing theory, $p_i = \rho^i(1-\rho)$, where $\rho = \lambda_1/\mu$. When burst starts, if the on-chip queue has i packets, the expected time to fill it is $\frac{L-i}{\lambda_2 - \mu}$. Therefore,

$$\begin{aligned} \Delta T^{\triangleleft} &= \sum_{i=0}^{\infty} p_i \frac{L-i}{\lambda_2 - \mu} = \frac{L}{\lambda_2 - \mu} - \frac{1}{\lambda_2 - \mu} \sum_{i=0}^{\infty} p_i i \\ &= \frac{1}{\lambda_2 - \mu}(L - \frac{\rho}{1-\rho}) = \frac{L - \psi}{\lambda_2 - \mu} \end{aligned} \quad (1)$$

where $\psi = \rho/(1-\rho)$.

Second, we compute $E(\Delta T^{\triangleright})$. $\Delta T^{\triangleright}$ is the time interval between a burst ends and the number of packets in off-chip queue is below w. Before the burst ends, the off-chip queue has received packets for $T_2 - \Delta T^{\triangleleft}$ and processed packets for $T_2 - \Delta T^{\triangleleft} - \Delta T'^{\triangleleft}$. Then the expected number of packets in off-chip memory when the burst ends is

$$(T_2 - E(\Delta T^{\triangleleft}))(\lambda_2 - \mu) + E(\Delta T'^{\triangleleft})\mu$$

Therefore,

$$\begin{aligned} E(\Delta T_2^{\triangleright}) &= \frac{(T_2 - E(\Delta T^{\triangleleft}))(\lambda_2 - \mu) + E(\Delta T'^{\triangleleft})\mu - w}{\mu - \lambda_1} \\ &= \frac{\lambda_2 - \mu}{\mu - \lambda_1} T_2 + \frac{\psi - w}{\mu - \lambda_1} \end{aligned} \quad (2)$$

Finally, $E(\Delta T'^{\triangleright})$ is the time that off-chip queue processes a queue of size while no packet arrives. It is easy to see $E(\Delta T'^{\triangleright}) = w/\mu$.

Taking all together, we have,

$$\begin{aligned} E(T_2'') &= T_2 - E(\Delta T^{\triangleleft}) + (E(\Delta T^{\triangleright}) + E(\Delta T'^{\triangleright})) \quad (3) \\ &= \frac{\lambda_2 - \lambda_1}{\mu - \lambda_1} T_2 - w\frac{\lambda_1}{\mu(\mu - \lambda_1)} - L\frac{1}{(\lambda_2 - \mu)} + C \end{aligned}$$

$$r_{off} = \frac{\frac{\lambda_2 - \lambda_1}{\mu - \lambda 1} T_2 - w\frac{\lambda_1}{\mu(\mu - \lambda_1)} - L\frac{1}{\lambda_2 - \mu} + C}{T_1 + T_2} \quad (4)$$

where $C = \frac{(\lambda_2 - \lambda_1)\psi}{(\mu - \lambda_1)(\lambda_2 - \mu)}$.

By equation (4), we have the following remarks regarding the two system parameters (on-chip queue length L and threshold w):

1) Larger size of on-chip queue L produces smaller r_{off}. This is because if larger L is, later the on-chip queue is full which postpones the start of off-chip queue, and in extreme case, the on-chip queue can handle a whole burst with its queue.
2) Larger threshold w produces smaller r_{off}. This is because the larger the w is, the more quickly packets are sent to on-chip queue for processing, which certainly decreases the use of off-chip queue. Yet, a too large will too early identify the end of a burst, and new packets will be sent to on-chip queue which will soon fill it and switch back to off-chip queue again, greatly complicating the operation.

C. Analyzing the On-chip Memory Size

To guide the implementation, we simply present a special case study of the system model. To simplify the analysis, we make an abstraction of the model, where we treat the off-chip queue and the on-chip queue as one uniform queue with infinite units. When the queue length is larger than L, we say the packet is stored in the off-chip memory. In this one-queue model, we use Poisson distribution to characterize normal traffic, and assume that the input speed is μ and the output is λ, respectively.

We denote by p_i the probability that the uniform queue has i packets. Therefore, the metric of the probability Pr_{off} that the packet is in off-chip memory is calculated with

$$Pr_{off} = \sum_{k=L+1}^{\infty} p_k \quad (5)$$

The number of processed packets and the number of arrival packets in unit time are both modeled via Poisson distribution, so $\lambda p_k = \mu p_{k+1}, k = 0, 1, \ldots$ and $p_k = \rho^k p_0$, where $\rho = \lambda/\mu$. Therefore,

$$Pr_{off} = \sum_{k=L+1}^{\infty} p_k = \sum_{k=L+1}^{\infty} \rho^k p_0 = \rho^{L+1}$$

When $\rho = \lambda/\mu = 0.9$, we depict the relationship between the length of the on-chip queue and the probability that the off-chip memory is activated as shown in Fig. 5. If we set the on-chip memory as 128 KB that could accommodate about 85 packets with maximum length of 1500 bytes, the probability to use the off-chip memory is in the order of 10^{-4}.

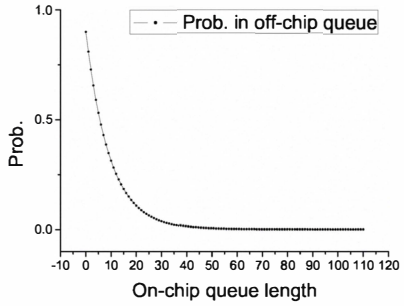


Fig. 5. The probability under different on-chip queue lengths

V. Experiment and Evaluation

The last two technical questions **Q3** and **Q4** in section I will be investigated in this section.

A. Testbed

The experimental environment is shown in Fig. 6. We use Altera Stratix IV FPGA development board [14] to implement the prototype of a router line-card as indicated on the right hand side of the figure. Besides the Altera Stratix IV E EP4SE530H35C2N FPGA, the development board provides one Gigabit Ethernet (GE) port, one 2-gigabytes (GB) DDR3 SDRAM DIMM with 72-bit data bus and one 72-Mb QDR II+ SRAM with 18-bit data bus. We develop the logic of the TM using Verilog HDL and download it into the FPGA chip on the board. On the left hand side of Fig. 6, there is a testing equipment from Spirent Inc. (AX/4000 Traffic Analyzer) [15], which is used to generate testing traces and to measure the processing delay.

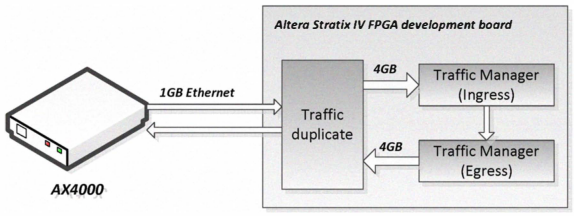


Fig. 6. Experimental environment

The traffic is injected to the development board through the 1-GE interface mentioned above. To test the performance under higher traffic rate, we scale the traffic rate by reproducing the packets inside the FPGA.

As mentioned earlier, the average link utilization is reported to be 40% or lower [4], so in our simulation, we set the maximal link utilization of the output direction in the TM to be 40%.

According to the theoretical analysis in Section IV, in our system, the on-chip memory size is 128KB.

TABLE I
THREE PACKET MANAGERS

Packet Manager	Structure
PM A	Traditional packet manager
DPM B	DPM with DQS
DPM C	DPM with class-based queue

We implement three versions of the Packet Manager Prototype systems supporting 4-GE throughput as shown in Table I. The traditional packet manager structure listed in the table supports DQS and always buffers the packets in its off-chip memory.

B. Experimental Traces

1) Synthesized Trace: We generate synthesized trace, which lasts for two minutes, with *Bi-modal uniform distribution.* (The time interval of the arriving flows is exponentially distributed with location parameter $\lambda = 1$). The packet sizes are uniformly distributed between 40B and 500B with probability 0.5, between 1000B and 1500B with probability 0.5. To simulate network congestion, the average link utilization of the input direction is set to be 40%, but the link utilization can reach up to 100% at random as shown in Fig. 7.

2) Real-life Trace: The CAIDA 2010 Internet Traces Dataset from "equinix-sanjose" OC192 monitors at 2010.3.25 (06:00am∼06:01am). The mean utilization is about 24.9115% [16].

3) Tester-generated Trace: With AX/4000 Traffic Analyzer, we can generate flows following the Markov Modulated Poisson process distribution. And the test datagram length is uniform distribution between 40B and 1500B. The average utilization of the input direction is set to be 40%, 30% and 20%, respectively. As described above, the generated data will be duplicated in the Altera Development Board shown in Fig. 6. As the superposition of two Poisson processes is still a Poisson process, the result of the duplication still flows the Poisson process distribution.

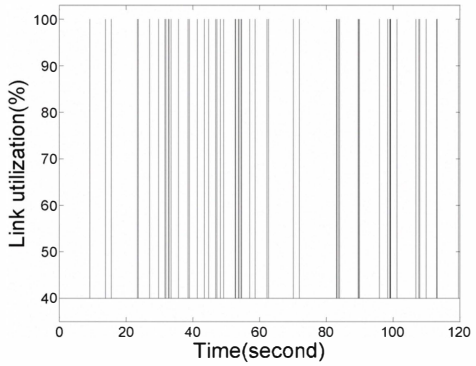


Fig. 7. The link utilization of the Synthesized Trace

C. Simulation Results

In this subsection, we use Modelsim, a simulation software tool [17], to simulate the TM prototypes and measure the power consumption, the latency and the buffer size of the prototype system.

By adopting different packet managers listed in Table I in the ingress and egress directions, we can implement three TM Prototypes as shown in Table II.

TABLE II
THREE TM PROTOTYPES

TM ID	Ingress	Egress
TM1	PM A	PM A
TM2	DPM B(per-flow)	DPM B(per-flow)
TM3	DPM C(per-class)	DPM C(per-class)

1) System Power: With the simulation results of the Modelsim, Quartus II software can calculate the average power consumption of the TM as indicated by Table III and Table IV, respectively. We can see that both the TM2 and TM3 have a considerable reduction in the power consumption, compared to the traditional TM1.

TABLE III
THE AVERAGE POWER CONSUMPTION OF SYNTHESIZED TRACE

TM ID	Average Power(mW)	Decrease Rate
TM1	9699.50	-
TM2	5286.56	45.5%
TM3	5203.00	46.4%

TABLE IV
SYSTEM POWER OF THE REAL-LIFE TRACE

TM ID	Average Power(mW)	Decrease Rate
TM1	9699.50	-
TM2	5150.21	46.9%
TM3	4968.36	48.8%

2) System Latency: The cumulative distribution function (CDF) of the system latency is shown in Fig. 8 for synthesized trace and, Fig. 9 for real-life trace. It shows that all the three TMs have the similar system latency. And under real trace, both TM2 and TM3, which always buffer packets in on-chip memory during the experimental period, have better delay performance than the TM1.

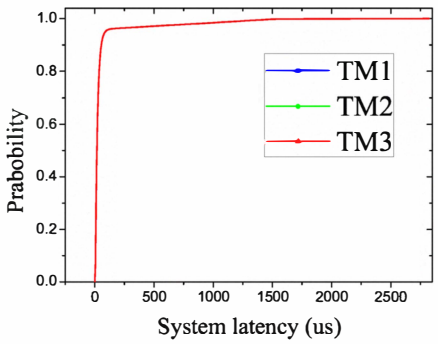


Fig. 8. The CDF of the system latency under synthesized trace

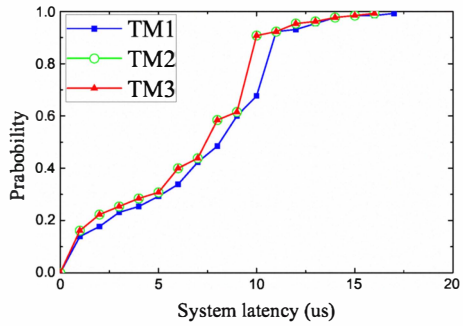


Fig. 9. The CDF of the system latency under real-life trace

3) Buffer Size of the Packet Manager: The buffer sizes of the three kinds of TMs are illustrated in Fig. 10 and Table V, which are almost the same. The main reason is that they adopt the same structure to support per-flow queuing. Although TM1 uses off-chip memory to buffer packets, the difference between the access latency of on-chip and off-chip memory is smaller than the time to receive one more data unit.

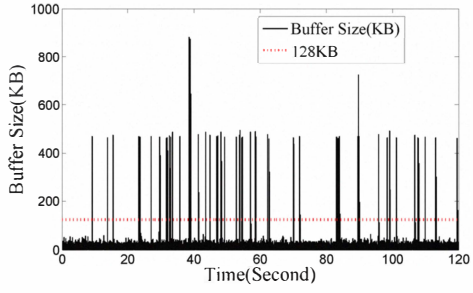


Fig. 10. The buffer size in the queue manager under synthesized trace

When the link utilization is about 40%, a quite few input packets (less than 128KB) are buffered in the queue manger, causing very low buffer utilization and small latency. As the congestion occurs and the average link utilization goes up to 100%, which is much larger than the maximal utilization in the output direction, the number of buffered packets in the queue manager increases. TM2 and TM3 activate the off-chip queue manager. But when the average link utilization becomes 40% again, the buffer size will quickly decrease to less than 128KB. So the analysis of buffer size to be 128KB in section IV is safe enough for the TMs with DPM.

TABLE V
QUEUE LENGTH OF THE REAL-LIFE TRACE

TM ID	Maximum buffer size(B)
TM1	58664
TM2	58664
TM3	58664

D. Experimental Results

In this subsection, we evaluate several metrics of the proposed DPM based on the real testbed, including peak power, dynamic power, latency and cost.

For only one DDR3 and QDRII external memory in the tested board can be used in the ingress or egress direction, we only implement DPM in ingress direction as shown in Table VI. Therefore, the power consumption of the actual TM is estimated as roughly twice of the measured power.

TABLE VI
THREE TM PROTOTYPES

TM ID	Ingress	Egress
TM1	PM A	Only on-chip
TM2	DPM B(per-flow)	Only on-chip
TM3	DPM C(per-class)	Only on-chip

1) Peak Power: The peak power is dissipated while the worst case traffic is injected into the system. To the DPM, if both on-chip and off-chip memories are activated to store packets, it consumes the largest power. In this case, the DPM logic as well as other control logic, one block of on-chip memory and one/two piece of off-chip memory will contribute to the peak power.

The peak power measured by Altera Stratix IV FPGA development board [14] is shown in Table VII. Compared with TM2, TM1 does not have extra DPM logic and on-chip memory, so its peak power consumption is a little bit less than TM2's. However, TM3 with class-based packet manager only uses one off-chip DDR memory, saving one QDR II+ SRAM, resulting in much less power consumption than TM1. From Table VII, we can calculate the peak power reduction to be (4113-2964)/4113×100%=27.9%.

2) Dynamic Power: Fig. 11 illustrates the comparison of the dynamic power consumption in a traditional TM with/without DPM.

The experimental results indicate that the proposed DPM can reduce dynamic power consumption by almost 37.5%. As the theoretical analysis results in Section IV-C demonstrated

TABLE VII
50% OF THE PEAK POWER

TM ID	Peak Power(mW)
TM1	4113
TM2	4132
TM3	2964

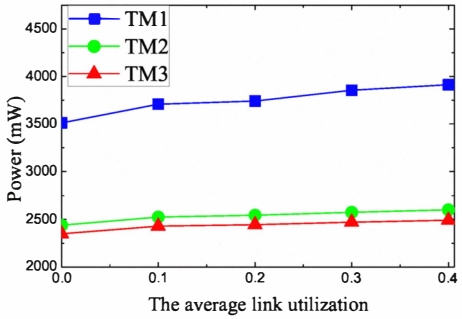


Fig. 11. 50% of the dynamic power in three TMs

that the probability to store packets in the off-chip memory is very small, the power consumption curve varies quite slightly with the link utilization. However, with the increase of the link utilization, the chance to activate the off-chip memory is enlarged. This explains why the power curve goes up as the utilization increases, but very slightly.

3) Latency: As shown in Table VIII, both the average latency of traditional TM and DPM increase stably with the increase of the link utilization. This phenomenon is easy to understand. When network traffic accumulates, the queue length in the packet manager grows, which leads to larger latency. The difference between the access latency of the off-chip memory and on-chip memory is very small, only serval nanoseconds, difference of the TMs' latency is also small as a result.

TABLE VIII
THE AVERAGE LATENCY OF THREE TMS

Average utilization	Average latency of TM1(us)	Average latency of TM2(us)	Average latency of TM3(us)
0.1	6.3	6.1	6.1
0.2	8.1	7.9	7.9
0.3	10.4	10.2	10.2
0.4	68.6	68.4	68.4

4) Cost: The above evaluations demonstrate the benefit of implementing DPM in TM. And now we will show how much extra cost DPM pays out to achieve the advantages. In Table IX, we illustrate the statistical results reported by the FPGA complier, Quartus II. The combinational ALUTs[3], the memory ALUTs of TM with DPM and Logic registers, are over two times of the traditional TM's, however, they only consume less than 1%, 3% and 3% of the total resources, respectively. Given TM3 requires only one off-chip memory, and saves one off-chip memory as well as the corresponding memory controller, as a result the demanding on-chip memory and registers are both less than that of TM1 and TM2.

[3]ALUT is short for adaptive look-up table

TABLE IX
SYSTEM COST

	TM1	TM2	TM3
Combinational ALUTs	4904(1%)	13888(3%)	11049(3%)
Memory ALUTs	189(<1%)	486(<1%)	486(<1%)
Dedicated logic registers	6856(2%)	16618(3%)	12744(3%)
Total registers	17420	17460	13464
On-chip Memory(B) /percentage	2422010 (11%)	2553632 (12%)	1738368 (8%)

VI. CONCLUSION

In this paper, we propose an energy-efficient TM architect called DPM. Compared with traditional designs, DPM has three advantages: first, the dynamic power usage is low, reduced by 37.5% on the prototypes tested with a small on-chip memory; second, the peak power usage is also low, reduced by 27.9% on the tested prototypes; third, DPM enables a reduction on packet scheduling delay, while also has a smaller memory and register footprint than traditional designs.

REFERENCES

[1] "Energy Use of Internet." http://uclue.com/index.php?xq=724
[2] G. Epps, D. Tsiang, and T. Boures, "System Power Challenges", 2006. http://www.slidefinder.net/c/cisco routing research/seminar august 29/1562106
[3] Carla Panarello et al., Energy Saving and Network Performance: a Trade-off Approach, 1st Int'l Conf. on Energy-efficient Computing and Networking, April 13-15,2010, University of Passau, Germany
[4] J. Guichard, F. L. Faucheur, and J.-P. Vasseur, Definitive MPLS Network Designs. Cisco Press, 2005.
[5] M. Gupta and S. Singh, "Greening of the Internet", in proceedings of SIGCOMM'03, 2003.
[6] Mingui Zhang, Cheng Yi, Bin Liu, Beichuan Zhang, GreenTE: Power-aware traffic engineering, ICNP2010, vol., no., pp.21-30, 5-8 Oct. 2010.
[7] Phillips, C., Singh, S., An Empirical Activity Model for WLAN Users, INFOCOM 2008, vol., no., pp.2065-2073, 13-18 April 2008.
[8] Alan Kennedy, Xiaojun Wang, Zhen Liu, and Bin Liu. 2008. Low power architecture for high speed packet classification. In Proceedings ANCS2008. ACM, New York, NY, USA, 131-140.
[9] Chengchen Hu, Yi Tang, Xuefei Chen, Bin Liu, Per-Flow Queueing by Dynamic Queue Sharing, INFOCOM 2007, vol., no., pp.1613-1621, 6-12 May 2007.
[10] Jindou Fan, Chengchen Hu, Bin Liu , Experiences with Active Per-Flow Queuing for Traffic Manager in High Performance Routers, ICC2010, vol., no., pp.1-5, 23-27 May 2010.
[11] Ye, T.T., Benini, L., De Micheli, G., Analysis of power consumption on switch fabrics in network routers, DAC2002, vol., no., pp. 524- 529, 2002.
[12] RFC4594. Configuration Guidelines for DiffServ Service Classes.
[13] Tian Pan, Chenhui Zhang, Xiaoyu Guo, Junchen Jiang, Ruian Duan, Tianhao Zheng, Chengchen Hu, Bin Liu. Accelerating Protocol Identification in High Speed Routers via Hardware Co-processor, in proceedings of IEEE INFOCOM 2011 Student Workshop.
[14] Stratix IV FPGA Development Board. http://www.altera.com/products/devkits/altera/kit-siv-gx.html
[15] Spirent AX4000. http://www.spirent.com/Solutions-Directory/AX_4000.aspx
[16] equinix-sanjose. CAIDA 2010. http://www.caida.org/data/monitors/passive-equinix-sanjose.xml
[17] Modelsim, Advanced Simulation and Debugging. http://model.com/