Power Optimization in Packet Switches through Reinforcement Learning

Advisor: Professor Daniel C. O'Neill
Divya Elayakumar

Problem statement

With the increase in network traffic, the packet switches operate at high speed to meet the high throughput rates. This fact, along with the increasing circuit densities make power a significant factor in the operation of the packet switches. When large number of packets are received and routed, the power required to operate the switches increases significantly. Power consumption is limited by physical constraints within the switch as well as the customer and industry standards. Scheduling algorithms that determines the packet to service must also balance throughput and power consumption. This necessitates power and delay aware scheduling algorithms for input queued packet switches.

Motivation

In 2007, United States Environmental Protection Agency (EPA) predicted that electricity use of data centers would get doubled in 2010.

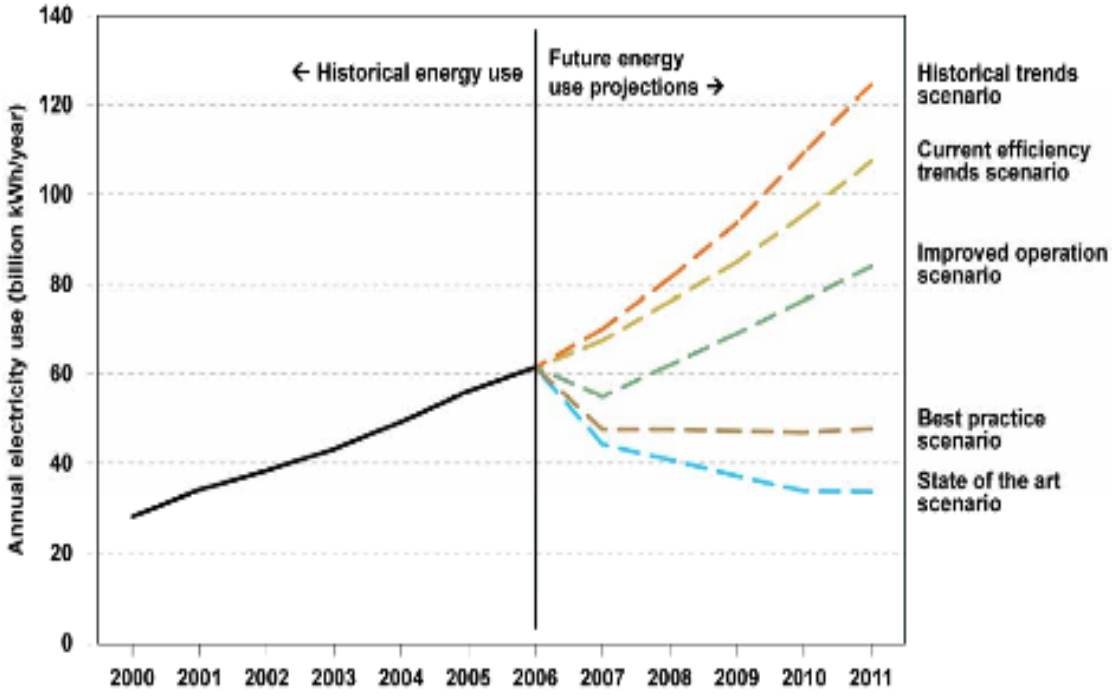

Figure 1. US EPA prediction of electricity use of data centers

Recent study by Professor Jonathan Koomey, Department of Civil Engineering, Stanford shows that electricity used by data centers worldwide has increased by 56% from 2005 to 2010. Electricity used by global data centers in 2010 accounts for 1.1-1.5% of total electricity use. In US , it is about 1.7-2.2%.

Several techniques are being developed and applied at system and circuit level to reduce the power consumption of the servers and storage elements. When the load is low, the devices can be turned off or operated in low power modes to reduce the power consumption. These kind of techniques have controlled the trend in electricity use that would have doubled otherwise.

Packet switches are the next major components where the power consumption is increasing with the increase in network speed, traffic demands and circuit density. The more the load is, the more is the power consumption. There could be factors such as leakage or static power contributing to this. In this work, we focus on the increasing dynamic power of packet switches.

Scheduling algorithm in packet switches determine the packets that can be serviced at a time. To reduce average power consumption,we can defer to service a packet at a time compromising performance. So, the scheduling algorithms must also strike balance between performance and power consumption. Hence we require power aware and delay aware scheduling algorithms that can provide trade off between performance and power.

Literature survey

The power and delay aware scheduling algorithm developed by Professor Daniel O'Neill et al. in (1) based on Linear Quadratic Regulation(LQR) selects service vector that best aligns with the pending switch backlog and then scales the switch performance to optimize the average power delay tradeoff for the switch. This two step algorithm sequence is repeated for every packet transfer period.

As explained in (1),the scheduling algorithm aims to minimize the two terms- power consumption and packet delay.The power consumption of the packet switch is $P = (1/2)\alpha CV^2 f$, C is the capacitance driven by the device, V is the voltage supply, f is the device clock frequency, α is the activity factor of the device.

Power consumption of a switch can be adjusted by varying the clock frequency of the device, varying its voltage supply and powering up or down the portions of the switch as necessary. Changing the supply causes a quadratic cost in power, while changing frequency or powering up/down causes linear cost. In general, if the cost is $F(s^t)$ as a function of service vector s^t , the power cost is the summation of $(s^t)F(s^t)$ over period of time.

Packet delay is measured by the backlog cost. A packet arriving in a longer queue will have a shorter delay than the packet arriving in a shorter queue. If the backlog vector is x^t , then the backlog cost is the summation of $(x^t)F(x^t)$ over period of time.

Hence scheduling algorithm intends to minimize the following function,
$J(x^t) = \Sigma \ ((s^t)F(s^t) + (x^t)F(x^t))$

In this work, we improvise the scheduling algorithm through reinforcement learning concepts. We need a scheduling algorithm that would dynamically learn from the load in the switches and minimize the average cost function.

Switch model

Consider a N X N switch that is composed of Line Card Processing units(LCP), virtual output queues(VOQ) and crossbar switching fabric.

It has N LCPs, one at each input. Each LCP has N VOQs one for each output. So, there are N^2 virtual output queues(VOQ). VOQs are defined as $VOQ_{i,o}$ where i refers to input port and o refers to output port. Each color in the VOQs in Figure 2 indicate the type of a packet.

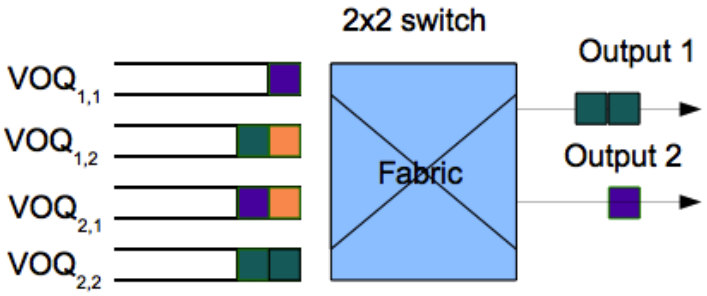

Figure 2. 2 X 2 switch model

Switch follows the crossbar constraint - no output simultaneously receives packets from more than one input VOQ and no input simultaneously transfers packets to more than one output. The packets that can be serviced at time t must follow this constraint.

Packet servicing is done at the beginning of time slot t , while packet arrivals occur at the end of time t.

Backlog state of the switch at time t is defined as the number of packets in each VOQ denoted as $x^t = [x^t_{1,}...x^t_{q}..x^t_{Q}]$, $Q = N^2$.

Packets are serviced at the beginning of a time t denoted by service vector $s^t = [s^t_{1,}...s^t_{q}..s^t_{Q}]$.

Packets arrive in the VOQs at the end of time t denoted by arrival vector $a^t = [a^t_{1,}...a^t_{q}..a^t_{Q}]$.

Backlog state of the switch a time t+1 is determined by the linear equation $x^{t+1} = x^t - s^t + a^t$.

Cost functions

Power and backlog cost equations defined in (1) are extended here.

Power cost in servicing packets at time t is quadratic and denoted by $(s^t)^T R\ s^t$, R is the power cost matrix.

Backlog cost at time t is quadratic and denoted by $(x^t)^T Q\ x^t$, Q is the identity matrix. The delay for a packet is proportional to the number of packets in the queue.

Total cost = $\Sigma\ \ ((s^t)^T R\ s^t + (x^t)^T Q\ x^t)$.

We need a scheduling algorithm that minimizes the average cost over a period of time.
The algorithm must continuously learn from the load in the switches and minimize the average cost. It can defer or service a packet at time t to minimize the average cost.

Approach

Reinforcement Learning

Reinforcement learning will be ideal for this problem. The agent interacts with environment and learns through consequences of actions-reward. An environment is represented by Markov Decision Process (MDP) consisting of four tuple (S,A,R,P), where S is state space, A is action space, R is reward model and P stands for transition model.

At time t the environment is in state $x_t \in S$, the agent takes an action $a_t \in A$ and receives a scalar reward r_t .The agent transitions to the next state x_{t+1}.

The agent and the environment generate a sequence of $x_0,a_0,r_0,x_1,a_1,r_1,x_2,a_2,r_{2...}$ The goal of RL agent, is to find a near optimal policy in a such a way that if the agent follows that policy from any given state then its sum of future reward is maximum .

Q-Learning

It belongs to the class of Temporal Difference methods. It is a simple, sample based, online and incremental reinforcement learning method. It does not require access to the model of the environment unlike dynamic programming.

It uses Q values defined as $Q(x,a)=E[r_0 + \gamma r_1+\gamma^2 r_2+\dots]$, γ is discount factor that weighs future rewards. The agent takes greedy actions mostly but also does exploration based on exploration policy to visit unseen states.

Algorithm

1.Initialize Q arbitrarily for all state-action pairs. The initial values can be set optimistically.
2.Choose proper (small) positive values for α_t .
3.Set discount factor γ , set it very close to 1, e.g. .99.
4.Repeat:
5.Take a_t from x_t according to a policy , and arrive at x_{t+1}.
6.Observe reward r_t.
7.Update Q-function for sample (x_t, a_t)

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t)+ \alpha_t [r_t +\gamma \max_{a'} Q_t(x_{t+1}, a')-Q_t(x_t, a_t)]$$

Power cost Estimation

We defined the power cost as $(s^t)^T R\, s^t$. To measure power cost, we need to find the power cost matrix R.

Network processors are programmable chips like general purpose processors but they are optimized for packet processing. They are programmable as a CPU but as fast as an ASIC.They perform functions such as pattern matching,key lookups, computation,bit manipulation,queue management etc. They are used in switches, routers, firewalls, VOIP bridges.

Our approach is to analyze a state of art network processor, its various design blocks and arrive at the values for cost matrix R. We have considered the Netlogic's network processor XLP832 for our analysis. Figure 3 represents the block diagram of XLP832.

XLP832 Architecture

XLP832 is a highly scalable network processor that supports key functions such as wired and wireless security,networking,storage,data center acceleration, load balancing and other acceleration engines. It is designed using 40 nm technology and offers processor core frequencies from 500 Mhz to 1.66 Ghz. It can support data rates from 10 – 160 Gbps.

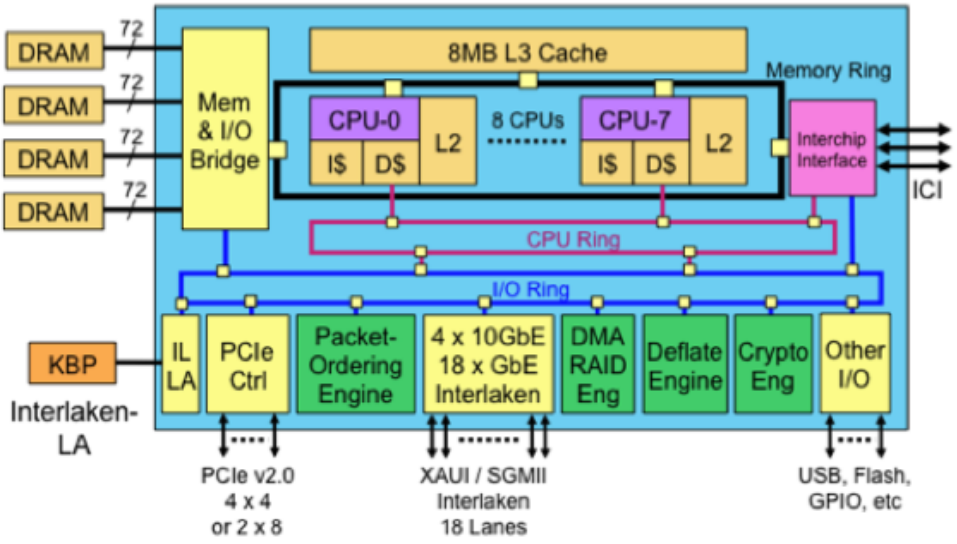

Figure 3. Netlogic XLP832 Block diagram

Processor core

The chip supports 8 cores, each core supports out of order execution, quad issue, 4-way simultaneous multithreading. The core architecture implements MIPS 64 Release -II ISA.

Multithreading enables processor to hide many of the clock cycles lost to data dependencies and other pipeline hazards by using instructions from other threads. When running single thread code, any thread can issue up to 4 instructions per cycle. When all the four threads are active, the processor acts like 4 single issue CPUs.

Figure 4 explains the various design blocks of the processor. The core has deeper 12 stage instruction pipelines. Apart from caches, it is composed of Instruction fetch,decode unit, seven functional units, load/store,retire unit , branch predictor, thread scheduler,rename registers and instruction queues. The thread scheduler determines which four instructions to fetch from which thread and then feeds the instructions into the queues.

Each ALU along with FPU has its own instruction queue. Two ALUs execute simple integer instructions and share their queue with load/store units. Another ALU executes simple integer instructions and branch instructions. The fourth ALU executes complex instructions such as multiply and divide. FPU is not used in the networking operations.

The core has 256 integer registers while there are 32 architectural integer registers. Renaming registers are helpful when a newer instruction reuses an architectural register that is still being used by an instruction in flight, renaming registers. The retire unit can support up to 100 instructions at a time.

With deeper pipelines, mis-predicted branches can inflict a greater penalty in flushing the pipeline. To overcome this, G-share and bimodal branch predictors are used. It also supports extended TLB and hardware page walker.

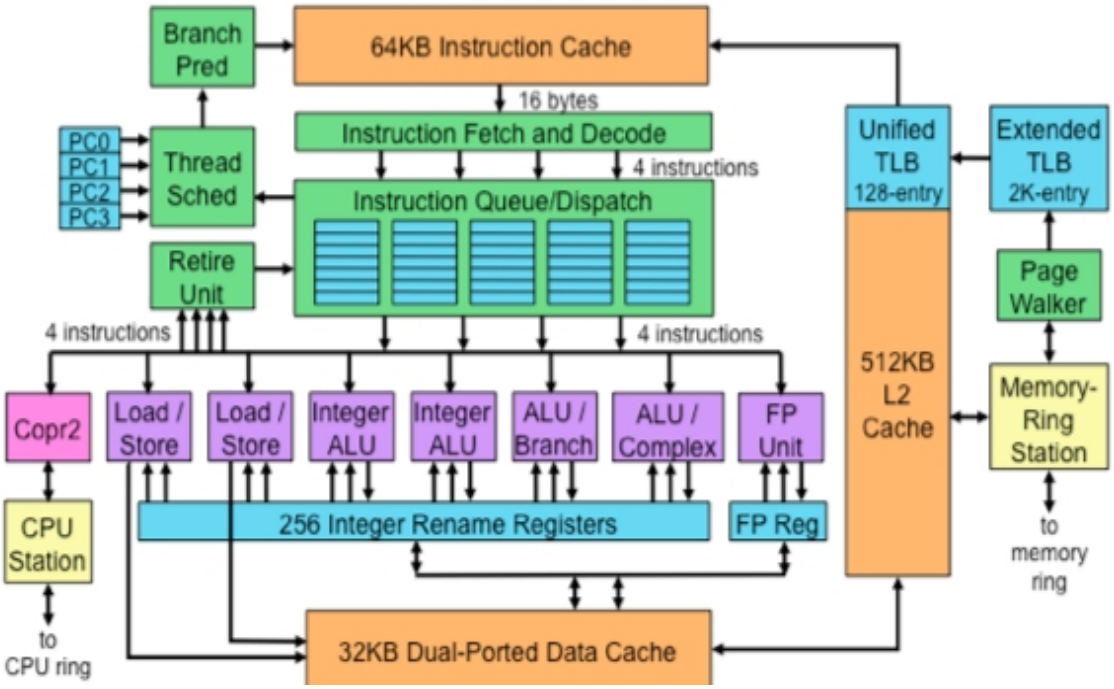

Figure 4. CPU block diagram

Ring Network

The chip has 3 rings- CPU ring connects the CPUs to each other, memory ring connects the CPU to memory controllers, I/O ring connects the CPU to I/O controller and acceleration engines. The rings are bidirectional and run at the core clock frequency. They support 64 bytes of data transfer per clock cycle and 40Tbps of bandwidth.

Network Accelerator Engine

This engine performs load balancing, it classifies packets depending on the processing required and allocates them to CPU on round -robin basis or by routing traffic to the CPU with the lightest load. It also does packet parsing, checksum generation and verification,TCP/IP/UDP checksum on both ingress and egress and TCP segmentation offload. It supports up to 40 Gbps of packet throughput. This engine is implemented using many(more than 40) 32 bit MIPS cores.

Security or cryptography engine

This engine provides 40Gbps of encryption/decryption. It is implemented using 32 bit MIPS core and cryptography cores. This engine is similar to the AU1550 security network processor developed by Netlogic. It operates at low frequency and supports special purpose logic or instructions for processing.

Packet Ordering Engine

This engine ensures packets are transmitted in the same order as received within a single flow while enabling simultaneous processing across multiple threads. This is also implemented using configurable cores.

DMA and storage acceleration/Deflate engine

An 8-channel DMA and storage acceleration engine with RAID 5 and RAID 6 are supported. This engine supports 10 Gbps of compression/decompression

Memory hierarchy
It supports MOESI coherent 3 level cache architecture- dedicated 64 KB I cache, 32 KB L1 D-cache, 512 KB 8 way L2 and 8 bank 16 way 8 MB L3. The chip has 4 on chip 72 bit DDR3 memory controllers supporting DDR3 1600 with 51.2 GB/sec bandwidth.

Packet flow

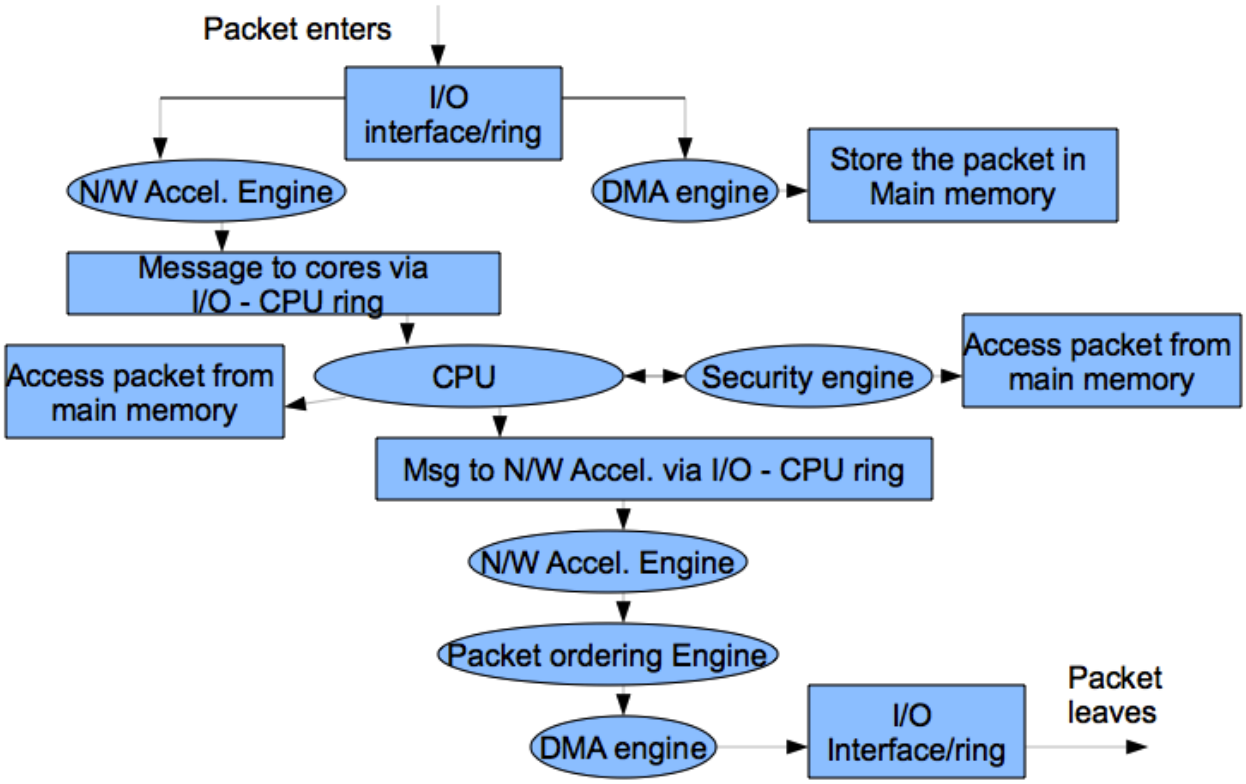

Figure 5. Packet flow in XLP832 chip

The packet enters the chip through I/O interface and reaches Network Acceleration and DMA engine.

While the DMA engine stores the packet in main memory, network acceleration engine parses the packet and determines its class and accordingly assigns it to a CPU for processing. Network accelerator also does checksum verification of the packet. The packet is stored in the main memory so that cores and acceleration engines can access it.

Network accelerator engine communicates to the CPU to start processing through I/O and CPU ring. The communication occurs through messages that carry packet information such as address where the packet is stored in main memory. This information is referred to as descriptors. All engines and cores access the packet through these descriptors and they communicate with each other through messages via rings.

The CPU fetches the packet from memory and processes it. For packets such as video that require encryption/decryption or compression /decompression, it invokes the security engine through rings.
The security engine accesses the packet and performs encryption/decryption. It then communicates back to the core through messages.

After completing its processing, core communicates to the network accelerator engine. Network accelerator performs checksum verification and provides control to the packet ordering engine.

Packet Ordering engine ensures that packets that entered the chip before the current packet are transmitted and then transmits this packet through DMA engine.

Power Calculation

Three types of packet are taken into account for this work and the following are the types and sizes of the packets,

- Ack – 48 bytes
- Data – 512 bytes
- Video – 1500 bytes

All these packet types undergo the packet flow discussed except that only video packet may require encryption/decryption.

CPU

The functions performed in the core are header check and bit manipulation in the payload. To perform these operations, the instructions required would be load, arithmetic and stores. Considering the packet sizes, the number of instructions required for this processing is calculated.

The relative power consumption of every block in the core is estimated so that power consumption in executing a ALU or load/store instruction can be determined. ALU complex and FP units will not be required for our processing. Using these values and the number of instructions required for every packet, power consumption for a packet processing is calculated.

Functional units	I-cache	IF/ID/branch prediction	Dispatch /thread scheduling	Integer ALU	Load/store/branch	CPU station(to CPU ring)	Integer register renames	Data cache	Retire unit	Total Power/ ALU instruction (nW)	Total power/LS instruction(nW)
Relative power values	0.14	0.3	0.1	0.01	0.22	0.02	0.11	0.04	0.06	0.72	0.98

Functional units	Instructions / packet	Power /packet(W)
ACK	18	1.608E-008
Data	93	8.308E-008
Video	18	1.608E-008

Figure 6. Power calculation for CPU

Security engine

This processing is required only for video packets. This engine is similar to AU1550 security network processor that consumes 500 mW/400 Mhz. Using this power consumption and data rate of 40 Gbps, the power consumption to process video packet of size 1500 B is calculated.

Packet type	Bytes to process	Time to process/packet(ns)	Power/packet (nW)
Video	1452	290.4	145.2

Figure 7. Power calculation for Security engine

Memory hierarchy
Power required per access at various levels are estimated which in turn is used to find power per byte access. This value is used to determine the power consumption in accessing a packet from memory.

As we move to higher levels of memory hierarchy such as L3 and main memory, memory latency is not so critical and so less sophisticated memory cells are used that would in turn consume less power.

Values from Freescale and Micron data sheet are used to extrapolate the values shown in Figure 8. Values highlighted in yellow are 'control knobs' in the calculation. They can be changed when required.

	% of total power	dynamic power (W)	power (W)/ access	power (mW)/byte accessed	accesses in 100 cycles	bytes accessed in 100 cycles	bytes/access
I cache	10.5%	14	0.14	8.75000	100	1600	16
dcache	3.0%	4	0.025	3.12500	160	1280	8
L2	4.2%	5.6	0.28	8.75000	20	640	32
L3	0.6%	0.8	0.8	6.25000	1	128	128

	Total power (mW)/ access	Read power (mW)/access	Activate power(mW)/access	Read power (mW)/byte accessed	bytes/access
Main memory	324	50	274	3.12500	16

Figure 8. Power calculation for caches and memory

Network Accelerator /Packet Ordering engine
Small in order 32 bit MIPS cores used in these engines consume power in the order of 100mW/1Ghz.

I/O ring
From Freescale data sheet, bus of width 106 operating at frequency of 100 Mhz with average transition of 2.7 Mhz consumes 17.8 mW. These values are extrapolated to obtained the power consumption of the I/O ring that can transfer 64bytes per clock. Assuming that it can transfer data on both edges of clock, the bus width is 256.

Freescale	Bus frequency (Mhz)	Bus width	Data transfers/clk	Data rate(MBps)	Avg. switching transition (Mhz)	Power of the bus (mW)
	100	106	2	2650	2.7	17.8
I/O bus	100	256	2	6400	2.7	42.98867925

Packet	Time to transfer(s)	Power/packet(mW)
ACK	7.5E-009	3.2242E-007
Data	0.00000008	3.4391E-006
Video	2.3438E-007	1.0075E-005

Figure 9. Power calculation for I/O ring

Final results
Design units that have same power consumption for all packets such as Network accelerator engine,packet ordering engine, rings are not included in the calculation.
Power consumption of DMA engine and memory controller are assumed negligible.

There are various control knobs in the calculation such as number of column elements in a row of DRAM, bytes/access, multi-issue count and frequency of the core etc. With these knobs, any change in the configuration or the values used for extrapolation can be easily updated.

Packet type	Network I/O bus/ring(W)	Packet storage in memory (I/O ring->Mem ctl & I/O bridge->main memory)(W)	Core processing (W)	Core accesses memory(L1-L2-L3-main memory) memory ring(W)	Security engine (encryption/ decryption) (W)	Security engine accesses the memory(W)	Access memory to transfer packet out (I/O ring->Mem ctl & I/O bridge->main memory)(W)	Network I/O bus(W)	Total power consumption/ packet (W)
ACK	3.2242E-010	0.424	1.608E-008	1.144	0	0	0.424	3.2242E-010	1.992000017
Data	3.4391E-009	1.874	8.308E-008	5.349	0	0	1.874	3.4391E-009	9.09700009
Video	1.0075E-008	4.9615	1.608E-008	1.144	1.452E-007	4.9615	4.9615	1.0075E-008	16.02850018

Figure 10. Total power consumption for packets

From the final power consumption values calculated above, we can see that memory accesses dominate the power consumption.

The following is the total power consumption values of the packets,
Ack - 2 W
Data - 9.1 W
Video - 16 W

Simulation

The simulation is done for 2 x 2 switch that has 2 input ports each with 2 VOQs and 2 output ports. Each VOQ can have at most 2 packets. No more than 2 packets can arrive at any time in a VOQ. Switch follows crossbar constraint and can service at most 2 packets from each port at a time. It supports 3 types of packets on each VOQ, but there can be be only packets of same type in a VOQ at a time.

MDP M= (S,A,R,P) that represents the switch is developed. There are 2401 states, each can have 25 actions.Transition probabilities that represent P are developed based on the linear equation,

$$x^{t+1} = x^t - s^t + a^t$$

Reward is represented by the total cost(power and backlog).The simulations are done in Matlab.

Dynamic Programming

Dynamic Programming(DP) is applied to validate average reward from Q-learning. Unlike RL, DP requires knowledge of the model - transition probabilities,P and reward,R, for a given state and action. RL needs to construct sample trajectories and learn from these samples while DP does not need to do this. P and R are not available in real world problems.

Value iteration algorithm

- For a state-action pairs, given R and P , Q-value update for iteration t is
  $Q_{t+1}(x, a) = \sum_{x'} P(x'|x,a) (R(x,a,x') + \gamma \max_{a'} Q_t(x', a'))$
- It synchronously updates Q values for all state-action pairs in a iteration.
- It repeats this until the Q values stop changing beyond a threshold.

Challenges

There were several challenges in the simulation due to the stochastic environment and large state space. The key issue was the difficulty in reaching all the states in the state space and the following are the techniques employed to address it,

1. Varying step size

Large step size initially(exploration phase) and small step size when the algorithm is about to converge(exploitation).

2 .Exploration policy

Greedy mostly, but also sometimes take random actions or actions that have not been hit frequently.

3. Transition probabilities

There can be any arrival pattern(not exceeding 2 packets in a port) at a time. With large state space and uniform probabilities for all the arrival patterns, it is harder to find optimal solution. So, we have skewed transition probabilities such that it is possible to cover the entire state space.

Results

It can be seen in Figure 11. that Q-Learning algorithm has converged closer to the optimal value found by value iteration algorithm.

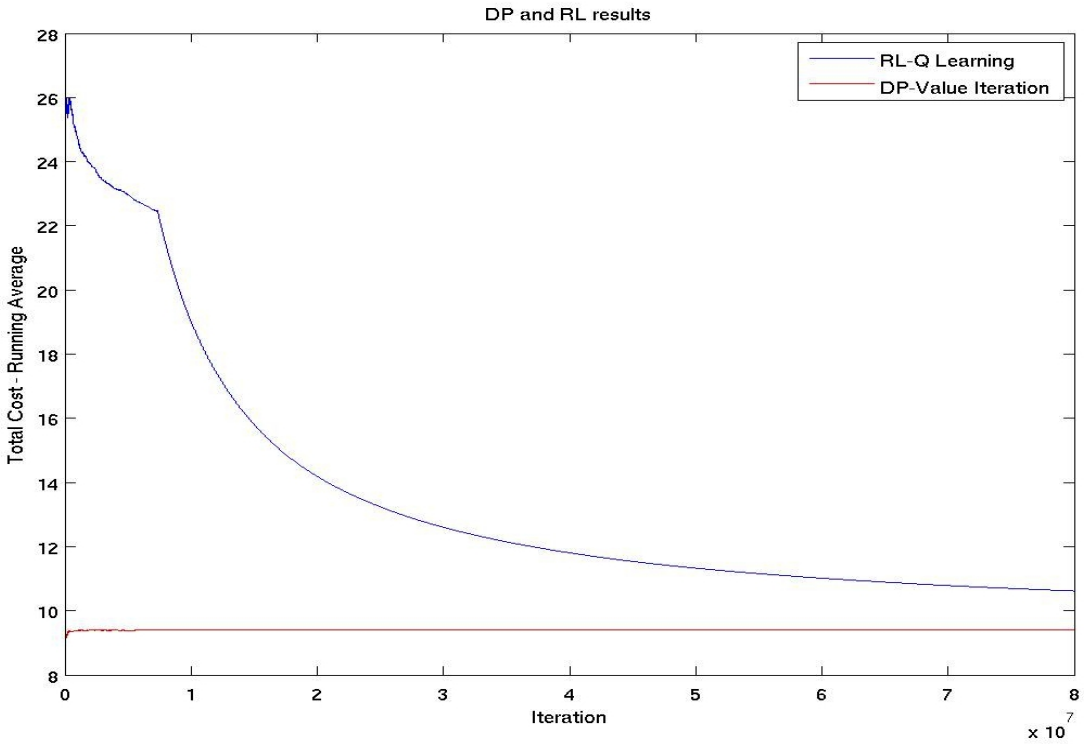

Figure 11. Simulation results of DP and RL

Acknowledgements

I express my sincere thanks to Professor Daniel C. O'Neill for his guidance and motivation throughout the work. I would also like to thank Hamid Reza Maei for his guidance in Reinforcement Learning and Wayne Yamamoto from MIPS Technologies for his guidance in the power estimation of XLP832 chip.

References

1. Lykomidis Mastroleon, Daniel O' Neill, Benjamin Yolken and Nicholas Bambos ,“Power and Delay Aware Management of Packet Switches” , IEEE Transactions on Computers, October 2011.
2. Jonathan G. Koomey , “ Growth in Data center electricity use 2005 t0 2010”, August 2011.
3. XLP832 Processor,Product brief, Netlogic Microsystems.
4. http://www.netlogicmicro.com/Products/ProductBriefs/MultiCore/XLP832.htm
5. “Netlogic broadens XLP family, Multithreading and four way issue with one to eight CPU cores” , Microprocessor Report ,July 2010.
6. “Specifying Power consumption ” ,Freescale Semiconductors. Document no : AN2436
7. “PowerPC MPC7455 I/O Power Evaluation” ,Freescale Semiconductors,Inc. .2004.
8. “RMI Alchemy AU1550 Processor,security network processor”, Raza Microelectronics,Inc..

9. US Environmental Protection Agency,“Report to Congress on Server and data center energy efficiency”, August 2007.
10. Stranger in an ARM world ,Ingenic designs MIPS CPU for JZ4700 Mobile Processor”,Microprocessor Report, 2012.
11. “Calculating memory system power for DDR SDRAM” Vol 10. Issue 2, Micron 2Q01.
12. Reinforcement learning book - http://webdocs.cs.ualberta.ca/~sutton/book/ebook