ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.2, February 2007, pp.229−235
DOI: 10.1360/jos180229

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

带输入队列并行交换的负载平衡分析*

董雨果[1], 汪胜荣[1+], 郭云飞[2], 刘 颖[1]

[1](工程大学 电讯工程学院,陕西 西安 710077)

[2](信息工程大学 国家数字交换系统工程技术研究中心,河南 郑州 450002)

Exploring Load Balancing of a Parallel Switch with Input Queues

DONG Yu-Guo[1], WANG Sheng-Rong[1+], GUO Yun-Fei[2], LIU Ying[1]

[1](Telecommunication Engineering Institute, Engineering University, Xi'an 710077, China)

[2](NDSC, Information Engineering University, Zhengzhou 450002, China)

+ Corresponding author: Phn: +86-29-84202040; Fax: +86-29-84202040, E-mail: ggsmart@gmail.com

Dong YG, Wang SR, Guo YF, Liu Y. Exploring load balancing of a parallel switch with input queues. *Journal of Software*, 2007,18(2):229−235. http://www.jos.org.cn/1000-9825/18/229.htm

Abstract: Parallel switch is an emerging switch technology by which we can build a high capacity switching system (such as a terabit or higher switch) from many small switch fabrics. This paper refers to the parallel switch with input queues as the Buffered Parallel Switch (BPS) and address the open issue of load-balancing for switch fabrics working parallelly and independently. Two classes of definition which depict the load balancing in different ways are proposed. Then conditions for BPS load balancing are analyzed and a family of distributed scheduling algorithms is presented. At last, a simple and efficient scheduling algorithm which can satisfy both classes of definition in a BPS without speedup is developed. Simulation results show the validity and performance of the load-balancing algorithm. Practical implementation of the distributed scheduling algorithms is also discussed.
Key words: parallel switch; load balancing; buffer; scheduling; distributed algorithm

摘 要: 并行交换是新兴的交换技术,基于该技术能够利用小型交换模块来构建大容量的交换系统,例如太比特或更高容量的交换机.把带输入队列的并行交换称为带缓存并行交换(buffered parallel switch,简称 BPS),重点研究其中并行且独立工作的交换模块之间的负载平衡问题.从不同角度出发,提出两种负载平衡的定义.基于两种定义,分别分析了 BPS 负载平衡的条件并提出分布式调度算法族.最后,提出一种简单而有效的调度算法,该算法能在无加速比 BPS 中同时满足两种定义,仿真实验结果表明了该算法的有效性和良好性能.另外,就算法的工程实现进行了讨论.
关键词: 并行交换;负载平衡;缓存;调度;分布式算法
中图法分类号: TP301 文献标识码: A

The future integration of mobile networks and Internet needs ultra-high performance equipments such as

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2001AA124011 (国家高技术研究发展计划(863); the Natural Science Foundation of Shannxi Province of China under Grant No.2005f17 (陕西省自然科学基金)
Received 2003-09-16; Accepted 2005-07-08

terabit or petabit switches. A novel architecture for high-speed switches is to make many gigabit switch fabrics work corporately[1−5], which is named as the parallel switching system (PSS). One of the critical issues for PSS architecture is load balancing of these switching fabrics.

A PSS with input queues to buffer high-line-rate packets is practical, to which we refer as a Buffered Parallel Switch (BPS). In this paper, We firstly introduce a model of BPS, and discuss the issue of load balancing for the BPS. We analyze the conditions for guaranteeing the load balancing, and then propose a family of distributed scheduling algorithms. Moreover, a basic and common scheduling algorithm is presented in order to be practically implemented. Simulations are carried out to evaluate our results. Finally, engineering considerations on implementing the distributed scheduling algorithms are discussed.

1 Definitions of Load-Balancing

The BPS model that we consider has N external output ports operating at rate R, as shown in Fig.1. The switching part is made up of K parallel switching elements, each of which is an $N\times N$ output-queued (OQ) switch with port rate r. The internal speedup S is defined by Kr/R. The buffer is arranged in each port as K equal size FIFO where each FIFO holds cells destined for the specific switching element[2,3]. Each input local scheduling part and each switching element are working independently, i.e., no information is shared among them. Some terms used throughout this paper are defined below. Timeslot: The time taken to transmit or receive a cell at the link rate of R. Layers: The center stage switches or the parallel switching elements. $A_{ij}(t)$: the aggregate flow traffic destined to the *j-th* output arriving to the *i-th* input of the switch within a timeslot interval $[0,t]$. $A^k_{ij}(t)$: the aggregate flow traffic dispatched to the *k-th* layer destined to the *j-th* output arriving to the *i-th* input of the switch within a timeslot interval $[0,t]$. $A_j(t)$: the aggregate flow traffic destined to the *j-th* output within an timeslot interval $[0,t]$. $A^k_{ij}(t)$: the aggregate flow traffic dispatched to the *k-th* layer destined to the *j-th* output within an timeslot interval $[0,t]$. $A_j(t)$: the aggregate flow traffic arriving to the *i-th* input within an timeslot interval $[0,t]$. $A^k_j(t)$: the aggregate flow traffic dispatched to the *k-th* layer arriving to the *i-th* input within an timeslot interval $[0,t]$. $Q^k_j(t)$: the length of the *j-th* output queue in the *k-th* layer within $[0,t]$. $IQ^k_i(t)$: the length of the queue ready to be dispatched to the *k-th* layer in the *i-th* input within $[0,t]$.

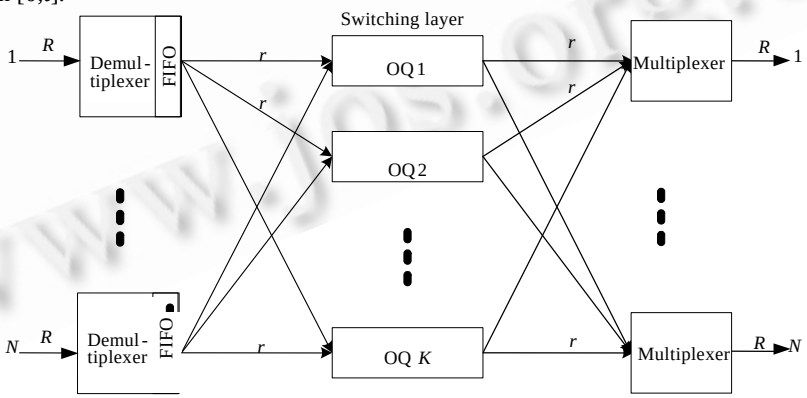

Fig.1 A model of buffered parallel switch

1.1 Weak load-balancing

Given a vector $X(t)=(x_1(t),x_2(t),...,x_N(t))$, we indicate with its Euclidean norm: $\|X(t)\|=\sqrt{\sum_{i=1}^{N}(x_i(t))^2}$. Since we can only know the past arrivals and the current state of a system in practice, Definition 3 in Ref.[6] is modified as: A system of queues is stable if $\lim_{t\to\infty}\|X(t)\|<\infty$. Given a BPS system, let $Q^k(t)$ be the row vector of the

output-queue lengths in the *k-th* layer at timeslot *t*, i.e., $Q^k(t)=(Q_1^k(t),Q_2^k(t),...,Q_N^k(t))$. These K vectors are independent and identically distributed. Let $IQ_i^k(t)$ be the single vector of the FIFO-queue length of the *k-th* FIFO in the *i-th* input at timeslot *t*. We know that if each parallel queue system has the equal arrival burden, the whole BPS system is load balanced. Similarly, we have the following definition of weak load-balancing.

Definition 1 (**weak load-balancing**). A BPS is in the state of weak load-balancing if the stability state of *K* layers is the same and the stability state of *K* FIFOs in any input is also the same.

1.2 Strong load-balancing

Let *Q*(*t*) be the queue length at timeslot t and R be the outgoing line rate of *Q*(*t*). Here we denote *Q*(*t*)/R as the processing time for the congestion queue *Q*(*t*), i.e. the delay time of the queue *Q*(*t*). Let us consider an *N*×*N* OQ switch as a reference switch that is work-conserving with port rate *R*. It is generally accepted to refer to such reference switch as ideal and stable[1−3]. Similarly, a reference queue system with line rate *R* is considered as being stable. We choose the following method to compare a BPS with a reference switch: comparison between the congestion processing time of FIFO and that of a reference queue system, and comparison between the congestion processing time of switching layer and that of a reference OQ switch, under identical external traffic conditions. Thus, we obtain the following definition.

Definition 2 (**strong load-balancing**). A BPS is in the state of strong load-balancing if the congestion processing time of FIFO is no more than that of a reference queue system, and the congestion processing time of switching layer is no more than that of a reference OQ switch, under identical external traffic conditions.

2 Analysis of Weak Load Balancing

The following theorem is for definition 1 and tells that which type of BPS traffic model can cause weak load balancing.

Theorem 1. A BPS is in the state of weak load-balancing if, for any *i*, *j*, *k* and timeslot *t*, the BPS satisfies:

$$A_{ij}^k(t)=\frac{1}{K}A_{ij}(t)+\Delta_{ij}^k \tag{1}$$

where Δ is a constant that does not depend on t, and $\sum_{k=1}^{K}\Delta_{ij}^k=0$.

Proof: The Euclidean norm of $IQ_i^k(t)$ is $\left\|IQ_i^k(t)\right\|=\sqrt{(IQ_i^k(t))^2}=\sqrt{(A_i^k(t)-rt)^2}$. We can write

$$\lim_{t\to\infty}\left\|IQ_i^k(t)\right\|=\sqrt{\lim_{t\to\infty}(A_i^k(t)-rt)^2}\ .$$

From (1) we can also obtain $A_i^k(t)=\frac{1}{K}A_i(t)+\Delta_i^k$. Then we have

$$\lim_{t\to\infty}\left\|IQ_i^k(t)\right\|=\sqrt{\lim_{t\to\infty}\left(\frac{1}{K}A_i(t)-rt+\Delta_i^k\right)^2}\ .$$

Since whether $\lim_{t\to\infty}\left\|IQ_i^k(t)\right\|$ is finite does not depend on *k*, for any input *i*, the stability of *K* FIFOs is the same. Similarly, we can conclude that stability of *K* layers is identical. Thus, by definition 1, weak load-balancing of the BPS is guaranteed.

We propose a family of scheduling algorithms for the BPS below. Let $P_K=\{(p_m,q_m), m=1,2,\ldots,M\}$ be a finite set of distinct integer pairs. Let each flow be arbitrarily partitioned at the input into blocks of cells having size q_m, and let the dispatch of consecutive blocks be performed independently. Out of each block of q_m cells belonging to the same flow, which arrives at the external input *i* and is destined to the external output *j*, no more than p_m cells are dispatched to the *k-th* layer of BPS.

The following theorem specifies the restriction of the scheduling algorithms for weak load balancing.

Theorem 2. The weak load-balancing of a BPS is guaranteed if, for any *i*, *j*, *k* and timeslot *t*, the family of traffic scheduling algorithms satisfies ($(p_m,q_m)\in P_K$):

$$p_m = \frac{1}{K} q_m + \Delta \tag{2}$$

Proof: If (2) is satisfied, $A_{ij}^k(t) = \sum_{m=1}^{M} p_m = \frac{1}{K}\sum_{m=1}^{M} q_m + \sum_{m=1}^{M} \Delta = \frac{1}{K} A_{ij}(t) + \Delta$. According to Theorem 1, the BPS is in the state of weak load-balancing.

3 Analysis of Strong Load Balancing

Theorem 3. A BPS is in the state of strong load balancing if and only if, for any *i*, *j*, *k* and timeslot *t*, the BPS satisfies:

$$A_j^k(t) \le \frac{r}{R} A_j(t),\ A_i^k(t) \le \frac{r}{R} A_i(t) \tag{3}$$

The family of scheduling algorithms for strong load-balancing is the same as that for weak load-balancing, except for the restriction which is specified in Theorem 3. Let us introduce Lemmas 1 and 2 firstly.

Lemma 1. Strong load balancing of the switching layers is guaranteed if for any *i*, *j*, *k* and timeslot *t*, the family of scheduling algorithms satisfies ($(p_m, q_m)\in P_K$):

$$\frac{p_m}{q_m} \le \frac{r}{R},\ \frac{p_m + 1}{q_m} > \frac{r}{R} \tag{4}$$

Proof: By (4), we can write $A_{ij}^k(t) / A_{ij}(t) = \sum_{m=1}^{M} p_m \Big/ \sum_{m=1}^{M} q_m \le (r/R) \sum_{m=1}^{M} q_m \Big/ \sum_{m=1}^{M} q_m = r/R$.

Then $$A_j^k(t) = \sum_{i=1}^{N} A_{ij}^k(t) \le \frac{r}{R} \sum_{i=1}^{N} A_{ij}(t) = \frac{r}{R} A_j(t).$$

By (4), we can obtain $A_j^k(t) + NM > \frac{r}{R} A_j(t)$. There must exist $A_j^k(t)$ and $A_j(t)$ that satisfy (3), i.e., (4) is sufficient.

Consider another family of scheduling algorithms for the input FIFOs: Out of each block of q_m cells belonging to the same flow, which arrives at the external input *i*, no more than p_m cells are dispatched to the *k-th* FIFO of input *i*. The restriction of the algorithms is described in Lemma 2, the proof of which is similar to that of Lemma 1. We do not prove Lemma 2 here for the reason of paper space limitation.

Lemma 2. Strong load balancing of FIFOs in each input is guaranteed if for any *i*, *j*, *k* and timeslot *t*, the family of scheduling algorithms satisfies (4).

By now we can easily see that a family of scheduling algorithms, which satisfy the requirements of both lemma 1 and 2, can guarantee strong load balancing of the BPS. Moreover, as Theorem 3 shows, we can simplify the restrictions.

Theorem 4. Strong load balancing of a BPS is guaranteed if for any *i*, *j*, *k* and timeslot *t*, the family of traffic scheduling algorithms satisfies: Out of each block of q_m cells belonging to the same flow, which arrives at the external input *i* and is destined to the external output *j*, no more than p_m cells are dispatched to the *k-th* FIFO of input *i*, provided $(p_m,q_m)\in P_K$.

4 Practical Scheduling Algorithm

In this Section, a distributed scheduling algorithm, named as UO-SA, for load balancing is proposed. In our simulations, UO-SA is compared with other typical algorithms to verify its delay performance.

4.1 Scheduling algorithm

Due to p_m and q_m being integer, algorithms of Theorems 2 and 4 are not convenient. Therefore we propose a practical one for uniforming output-port scheduling algorithm (UO-SA), which is implemented independently in every input demultiplexer.

UO-SA: For any *i*, *j*, *k* and timeslot *t*, out of each block of *mK* cells belonging to the same flow, which arrives at the external input *i* and is destined to the external output *j*, exactly *m* cells are dispatched to the *k-th* FIFO of input *i*, i.e.

$$p_m = \frac{1}{K} q_m \quad ((p_m, q_m) \in P_K) \tag{5}$$

The following theorem clarifies the relation between UO-SA and weak load balancing. In (2), let Δ be 0, and then we get (5). So Theorem 5 is obvious.

Theorem 5. UO-SA can guarantee the weak load balancing of a BPS.

Theorem 6 tells the relation between UO-SA and strong load balancing, under the sufficient condition of internal speedup *S*=1 which is common and practical in high-speed switch architecture (note that Theorem 5 has not this restriction).

Theorem 6. UO-SA can guarantee the strong load balancing of a BPS with speedup *S*=1.

Proof: Since *S*=1, we have $K = \frac{R}{r}$. Then we obtain $\frac{m}{mK} = \frac{1}{K} = \frac{r}{R}$ and $\frac{m+1}{mK} = \frac{1}{K}\left(\frac{m+1}{m}\right) = \frac{r}{R}\left(1+\frac{1}{m}\right) \geq \frac{r}{R}$. So (m,mK) satisfies (4) i.e. $(m,mK) \in P_K$.

From Theorems 5 and 6, it is easy to see that UO-SA can guarantee both weak and strong load balancing while there is no speedup, i.e. *S*=1. There is a very simple type of UO-SA[2,3]: In every input demultiplexer, for each output-port arrived, cells are uniformly distributed to each FIFO in a Round-Robin way. This method will be used in our simulations below. Since delay time is one of the most critical key in the study of switch architecture and OQ is accepted as an ideal switch, we refer load balancing as strong load balancing in the rest of this paper.

4.2 Simulation method

We carried out simulations for a deep understanding of UO-SA. The BPS model that we studied has 8 ports with line rate 160Gbps, 8 switching layers with port rate 20Gbps and a cell length of 64 byte. Its internal speedup is *S*=1. Three types of scheduling algorithm for load balancing were simulated to compare their delay performance. Random scheduling algorithm (RSA); Uniform scheduling algorithm (USA); UO-SA.

We compared three algorithms in different loaded arrivals. Simulations were run for cell arrivals with Bernoulli Process. For Bernoulli process, the simulation run time was 1 million cell times. The offered load was varied from 10% to 100%. The output port destinations were selected as high-degree balanced (HB), low-degree balanced (LB), and low-degree unbalanced (LU)[7].

4.3 Simulation results

In Fig.2, we can see that under HB Bernoulli arrivals, a BPS with RSA becomes unstable above 65% load while UO-SA can hold load-balancing even on 100% load. The reason is simple. RSA distributes most of the cells with line rate 160Gbps to the FIFO with output rate 20Gbps, which leads to a very heavy congestion in port queues. Moreover, RSA is likely to dispatch cells of the same output port to only one switching layer, which causes a great congestion in OQ switching fabrics. USA has a better performance than UO-SA under 50% load but above 80% BPS with USA becomes unstable. USA distributes cells uniformly to each FIFO in any input, which decreases delay time of BPS port but highly increases the delay in switching fabrics, while UO-SA just causes a bounded congestion

in FIFO[2,3]. Moreover, UO-SA can guarantee load balancing under any arrival situation.

Figure 3 shows the comparison of load-balancing on different scheduling algorithms under LB Bernoulli arrivals. Under LB Bernoulli, the BPS with RSA becomes unstable above 40% load and USA can hold load-balancing below 70%. The reason is that HOT-SPOT traffic is more likely to happen with LB than with HB, which means that for some duration, several inputs have the same destined traffic. UO-SA always has stable performance from 10% to 100% loads because it can distribute HOT-SPOT traffic uniformly to every switching layer. Comparison under LU Bernoulli is shown in Fig.4, which is like Fig.3 except that the unstable deadlines of BPS with RSA and USA are lower.

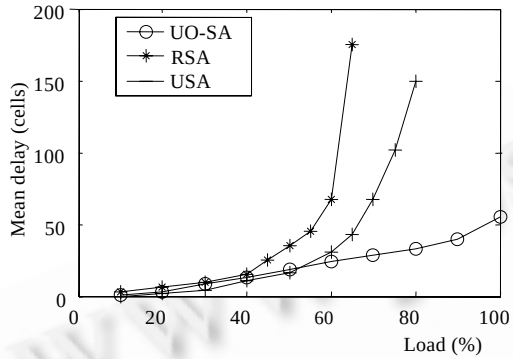

Fig.2 Comparison under HB Bernoulli

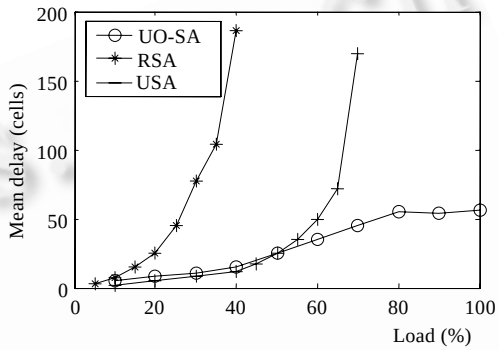

Fig. 3 Comparison under LB Bernoulli

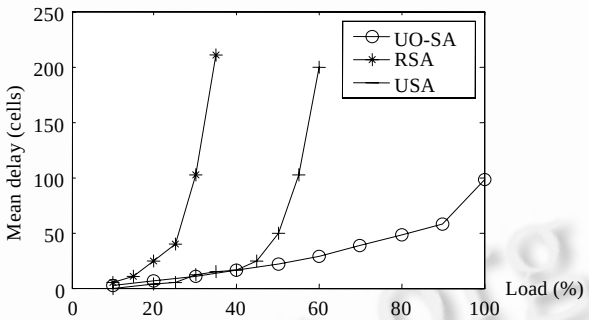

Fig.4 Comparison under LU Bernoulli

5 Practical Implementation

Where and how to run load-balancing algorithms? The scheduler is in inputs. In a normal PPS without input queues the arrived cells are directly dispatched to switching layers, while in a BPS, cells are buffered in FIFOs and waits to be sent to switching layers later. In a high-speed BPS, load-balancing algorithms must be executed in a distributed and independent way, i.e., each input has an independent scheduler. Moreover these schedulers have no feedback from switching layers in order to decrease the complexity of internal control communication.

Will the load-balancing algorithms be implemented in such high-speed line rate? As we discuss above, SQ-UA(m) can improve the delay performance of a BPS but it simultaneously increase computation burden of schedulers. While implementing load-balancing algorithms in the high-speed environment, a proper balance between performance and computing complexity is a key issue we should consider. However, both UO-SA and SQ-UA can be implemented in simple hardware. Reference [5] reported that an algorithm with the same computing complexity as SQ-UA had been implemented in CMOS process in a 320Gbps switch fabric in less than 10ns. Due to

the space limitation of the paper, we will not describe the details of the implementation.

How to guarantee the cell order in a BPS under load balancing? In high-speed switches, arriving packets with variable length are chipped into cells for fast switching. After being switched, cells need to be assembled. Keeping cell order is one of the difficult problems which have highly baffled the realization of parallel switches. Some mechanisms, with lack of detailed analysis and performance evaluation, were mentioned in Refs[3,5] to maintain cell order in parallel switches. Therefore, we proposed VIQ (virtual input queue) architecture and RRSK (round-robin for sequence keeping) algorithm[8]. Theoretical analysis and simulation results show that the average delay of a BPS can be less than that of OQ plus a constant through our mechanism for maintaining cell sequence.

6 Conclusions

Due to the commercial needs of scalability and technical limitations of current chip, it seems to be a trend of switch technology to build a terabit or petabit switch from many small switching fabrics, such as multi-stage switches, hypercube switches and parallel switches. We focus on sufficient conditions for BPS load-balancing in this paper. In the proposed model of BPS, weak load balancing and strong load balancing are defined. Traffic relations and a family of scheduling algorithms for each class of load balancing are separately analyzed. A basic algorithm belonging to both algorithm families, which is called UO-SA, is proposed and simulated. Extensions to UO-SA and some topics on implementing distributed scheduling algorithms in the BPS are presented. Future work will focus on details of UO-SA extensions and implementation.

References:

[1] Iyer S, Awadallah A, McKeown N. Analysis of a packet switch with the memories running slower than the line rate. In: Proc. of the INFOCOM 2000. 2000. 529−537.

[2] Iyer S, McKeown N. Making parallel switches practical. In: Proc. of the INFOCOM 2001. 2001. 1680−1687.

[3] Iyer S, McKeown N. Analysis of the parallel packet switch architecture. IEEE/ACM Trans. on Networking, 2003,11(2):314−324.

[4] Khotimsky DA, Krishnan S. Stability analysis of a parallel packet switch with bufferless input demultiplexors. In: Proc. of the IEEE ICC 2001. 2001. 100−111.

[5] Wang W, Dong LB, Wolf W. A distributed switch architecture with dynamic load-balancing and parallel input-queued crossbars for terabit switch fabrics. In: Proc. of the INFOCOM 2002. 2002. 212−223.

[6] Leonardi E, Mellia M, Neri F, Marsan MA. On the stability of input-queued switches with speed-up. IEEE/ACM Trans. on Networking, 2001,9(1):104−118.

[7] Goudreau M, Kollioulos S, Rao S. Scheduling algorithms for input-queued switches: Randomized techniques and experimental evaluation. In: Proc. of the IEEE INFOCOM 2000. 2000. 1634−1643.

[8] Dong YG, Wang BQ, Guo YF, Wu JX. Maintaining packet order for the parallel switch. In: Proc. of the GCC 2003. 2003. 176−179.

DONG Yu-Guo was born in 1976. He received his Ph.D. degree from the Telecommunication Engineering Institute, Engineering University. His current research areas are high-speed networks, etc.

GUO Yun-Fei was born in 1963. He is a professor at the Information Engineering University and a CCF senior member. His current research areas are communication technology, etc.

WANG Sheng-Rong was born in 1977. He is a Ph.D. candidate at the Telecommunication Engineering Institute, Engineering University. His current research areas are computer networks, etc.

LIU Ying was born in 1980. She is a Ph.D. candidate at the Telecommunication Engineering Institute, Engineering University. Her current research areas are computer grids, etc.