Implementing High Throughput Contention-Tolerant Crossbar Switch

Zhiyi Fang[1], Shuaibing Lu[1], Guannan Qu[1], Jianfei Zhang[2], and S. Q. Zheng[3]
[1] College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2] School of Computer Science and Technology, Changchun University of Science and Technology, Changchun 130012, China
[3] Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083, USA
Email: fangzy@jlu.edu.cn; lushuaibing11@163.com; gnqu@jlu.edu.cn

Abstract— Recently, an innovate switch architecture named Contention-Tolerant Crossbar switch, CTC(N), was proposed. Without resolving output contentions, the controllers are able to fully distributed in CTC(N). It largely reduces the scheduling complexity. However, It has been proved that the saturated switch throughput is bounded by 63% without any scheduling algorithms. In this paper, we present an implementation scheme named Two-Stage Contention-Tolerant Crossbar, denoted as TCTC(N, k). TCTC(N, k) uses Contention-Tolerant Crossbar as its basic switch component. And we will theoretically prove that TCTC(N, k) achieves high throughput with small size CTC components and without complex hardware and internal speedup.

Index Terms—Contention-Tolerant, switch, queueing analysis

I. Introduction

Switch, as the core part of switches and routers, delivers the packages (cells) arriving at input ports to their targeted output ports. Since the simpleness, crossbars have been widely used in commercial switches. In crossbar switches, input ports and output ports are connected by controlling the states of crosspoints of crossbar to be cross or bar state. Multiple input ports have cells intending to the same output ports, however, an output port only receives one cell in a time slot without speedup, and all others have to be remain in input port buffers. How to resolve the output contentions and optimize performance using scheduling algorithms with or without speedup has been a hot topic dozens of years. In order to overcome the Head-of-Line (HoL) blocking, buffers in input port are arranged as Virtual Output Queue (VOQ). Maximum scheduling algorithms are able to guarantee optimal performance by operating on maximum size or weight matching, e.g. [1], [2]. However, centralized scheduler operating need at least $O(N^{2.5})$ time complexity. It is hard to satisfy high speed and large scale network. Iterative heuristics for finding maximal matching were considered instead, which are usually implemented as 2N arbiters cooperating with iterative RGA (Request-Grant-Accept) signal exchange. The representative works are PIM [3], iSLIP [4], and so on. It has been proved that $O(logN)$ iterations are required to obtain an maximal matching. Although implemented in hardware, these schedulers are considered too slow with very high costs for high-speed networks. In addition, for resolving output contentions and achieving high performance, all input and output ports are involved in scheduling process with conventional crossbar. It limits the scale of switch on single chip even using VLSI techniques.

In order to reduce scheduler complexity, a small buffer was introduced to each crosspoint of crossbar. Such switch is called buffered crossbar switch. The scheduling process of buffered crossbar operates in two phases. In the first phase, each input port selects a cell to place into a crosspoint buffer in its corresponding row, and in the second phase, each output port selects a crosspoint in its corresponding column to take a cell from. Input (resp. output) ports operate independently and in parallel in the first (resp. second) phase, eliminating a single centralized scheduler. Crosspoint buffers are used as a decoupling mechanism for implementing separated distributed and parallel input scheduling (first phase) and output scheduling (second phase). Some works on buffered crossbar switches with or without internal speedup include, for example, [5]-[9]. The cost of crosspoint buffers, which requires at least $O(c \cdot N^2)$ memory space, where c is the number of bits in a cell, is used to trade for reduced control complexity. And, crosspoint buffers and the circuit for schedulers take a large chip area, which also severely restricts the scalability of buffered crossbar switches.

Recently, we propose an innovate switch architecture called Contention-Tolerant Crossbar Switches (*CTC*(N)) [10]. *CTC*(N) is able to tolerate output conflict automatically, thus the schedulers are fully distributed over inputs, avoiding central control or signal exchange. It largely reduces the scheduling complexity. In this paper, we will present a two-stage CTC architecture called *TCTC*(N, k). *TCTC*(N, k) is implemented with small size CTC as its basic switch component, and significantly reduces crosspoint complexity. By analyzing the queueing model of *TCTC*(N, k), we will prove that it achieves high switch throughput with k = 2

Manuscript received May 8, 2015; revised September 20, 2015.
This work was supported by the International Cooperation Project of Jilin Province of China under Grant No. 20130413054GH.
Corresponding author email: gnqu@jlu.edu.cn.
doi:10.12720/jcm.10.10.791-796

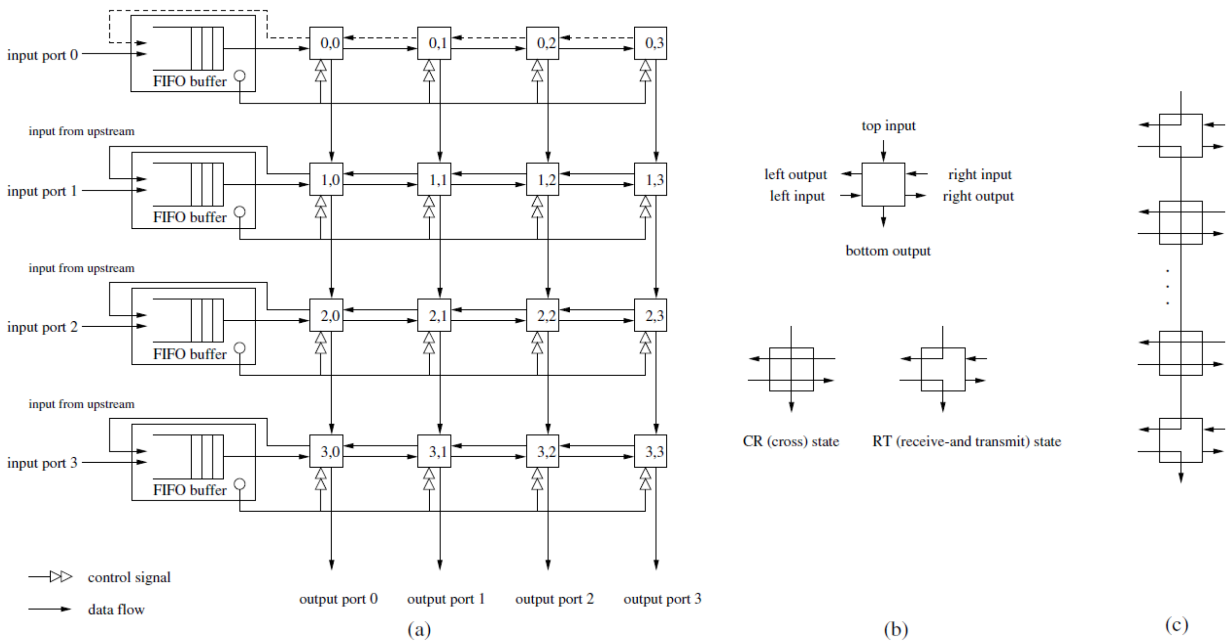

Fig. 1. (a) *CTC*(*N*) architecture with $N = 4$. (b) A crosspoint SE and its two states. (c) Each column of *CTC*(*N*) can be considered as a reconfigurable bus.

II. Related Work

The fabric of *CTC*(*N*) comprises N^2 crosspoints arranged as a *N*-by- array, as shown in Fig. 1(a). Each crosspoint is a Switch Element (SE), and the SE at row *i* and at column *j* is denoted as $SE_{i,j}$, who has two states, i.e. Cross state (CR state) and Receive-and-Transmit state (RT state), as shown in Fig. 1 (b).

The SEs are initialized to be CR states, and are controlled by input ports in a synchronized fashion. Each input port is equipped with a scheduler. At the beginning of each time slot, if input port *i* wants to transmit a cell to output port *j*, it sets $SE_{i,j}$ to RT state with all other SEs in row *i* remaining in CR state; Otherwise, all SEs in row *i* remaining in CR state. In this way, the output column is dynamically partitioned into several segments so that parallel cell transmissions are performed on these segments concurrently, as shown in Fig. 1 (c). It is possible that multiple input ports send cells to the same output port *j* during a time slot. In such a case, all cells but one are intercepted and are buffered by a downstream input port, with the cell from the lowest input port transmitted to output port *j*. Output conflicts in column *j* are automatically avoided without losing cells. Without resolving output conflicts, the *N* schedulers are fully distributed over input ports, and operate independently with zero knowledge of other input ports. These attractive properties make *CTC*(*N*) more scalable than conventional crossbars.

In [10], we also developed a mathematic model of CTC(N) using queueing theory and analyzed the existing issues of the CTC architecture: 1. Without internal speedup, the saturated throughput, i.e. throughput under full offered load, decreases with the increasing of switch size. The saturated throughput of CTC(N) with FIFO single queue is about 63% in the worst case.2. More downstream inputs suffer from more overloads, which lead to reduction of throughput. 3. Cells from upstream inputs would be intercepted by downstream inputs. Larger number of downstream inputs causes longer worst travel path for cells. In this case, cells might suffer from out-of-sequence problem. Our subsequence work discussed those issues by presenting improved architectures and scheduling algorithms.

High throughput is able to achieve by using sophisticated scheduling algorithm. Reference [11] proposed a fully distributed scheduling algorithm called Staggered Polling (SP for short). With this algorithm, the queues in each input port are arranged as N FIFO queue, one for each output port, called Virtual Output Queue (VOQ). The schedulers are composed by two sub-schedulers, i.e. primary scheduler and secondary scheduler. The primary scheduler in each input port chooses a specific output queue to server in round-robin pattern, and different output queue will be served by different scheduler. In this way, interceptions in output line can be avoided. While the output queue which should be served by a primary queue is empty, the corresponding secondary scheduler will choose one non-empty output queue to server under some preset scheduling strategy. Using this scheduling algorithm, high performance achieved under Bernoulli i.i.d. uniform traffic. Under bursty traffic, however, it didn’t perform well.

In order to increase the performance, we discussed several improved architectures [10], [12]-[14]. In [10], [12], we proved that 100% throughput achieves with two planes of *CTC*(*N*) or with speedup two. In addition, a queue model was developed to analyzed the cell delay in *CTC*(*N*), and the mathematical result were proved its correctness by simulation results [12]. In [13], we presented a delicate version of *CTC*(*N*), named DiaCTC. By rearranging the crosspoints only, it is able to achieve high performance with SP scheduling algorithms without any change. Article [14] proposed an parallelized version of *CTC*(*N*) named *PCTC*(*N*). In *PCTC*(*N*), the entire fabric can be divided into several regions. Those regions operate independently and in parallel, which highly improve the performance.

Since cells from upstream input ports might be intercepted and buffered by downstream input ports, *CTC* architecture suffers from cell out-of-sequence problem. We discussed this issue in [15], [16]. In [15], a fully distributed scheduling algorithm called SELF-ADJUSTED scheduling algorithm was proposed. With this algorithm, each input port has an independent scheduler, and the queues are arranged as N VOQs and an Upstream Queue (UQ). If the UQ is non-empty, which means that cells are from upstream input ports still exist, the scheduler will choose the UQ to serve, otherwise, VOQ will be served in Round-robin pattern. In this way, cells will arrive at output port in their original order. In [16], we developed an analytical model named Multi-level Contention-Tolerant Crossbar, denoted as *MLCTC*(*N*). It simplifies the queueing behavior, and can be described mathematically as an open queueing network systems. Simulations results prove the correctness of *MLCTC*(*N*). And, we discuss the speedup parameter of *CTC*(*N*) matching the OQ switch using *MLCTC*(*N*).

Contention-Tolerant crossbar switch architecture opens a new space to design switches and leaves lots of challenges to overcome as well. Even we have discussed several issues and improvements, those challenges could be overcome in different directions. In this paper, we will consider to implement high throughput CTC in an innovative way.

III. Two-Stage Contention-Tolerant Crossbar Switch

In this paper, we introduce an implementation scheme of *CTC*(*N*) called Two-stage Contention-Tolerant Crossbar Switch, denoted as *TCTC*(*N*, *k*), where *N* is the input/output ports number of the switch, and *k* is the input/output number that each single switch module has, $2 \le k < N^{\frac{1}{2}}$. *TCTC*(*N*, *k*) is composed by input stage and output stage, each of them has m modules, where $m = N/k$. Each input module (output module) is a $k \times m$ ($m \times k$) Contention-Tolerant Crossbar. The d^{th} input module (output module) is represented as IM_d (OM_d). I_i, the i^{th} input of *TCTC*(*N*, *k*), is the $(i\%k)^{th}$ input of $IM_{i/k}$, and O_j, the j^{th} output of *TCTC*, is the $(j\%k)^{th}$ output of $OM_{j/k}$. The p^{th} output of IM_q is connected to the q^{th} input of OM_p. Let $c_{i,j}$ is a cell arriving at I_i with O_j as its output destination. It is queued in input buffer at $IM_{i/k}$ when it arrives at *TCTC*(*N*, *k*). It will be sent to $OM_{j/k}$ by the $(j/k)^{th}$ output of $IM_{i/k}$, and will be queued in the $(i/k)^{th}$ input buffer at $OM_{j/k}$, waiting for being forward to its output destination, i.e. O_j.

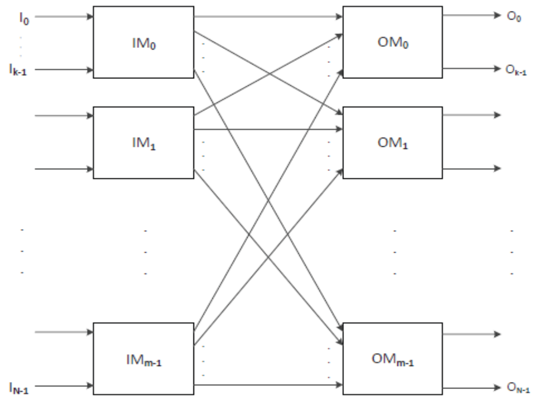

Fig. 2. Two-stage contention tolerant crossbar switch

In the worst case, a cell being transmitted from input 0 (the top input port) might be intercepted $N - 1$ times in *CTC*(*N*). However, in *TCTC*(*N*, *k*), the worst interception time is $2(k - 1)$. Some examples are shown in following table. Cell being intercepted would cause long delay and out-of-order length. *TCTC*(*N*, *k*) significantly release this problem in this way.

TABLE I: THE WORST INTERCEPTION TIMES IN *TCTC*(*N*, *K*)

	N=64	*N*=128	*N*=1024
k=2	2	2	2
k=8	14	14	14
k=16	30	20	30
CTC(N)	63	127	1023

IV. Throughput Analysis of TCTC(N, K)

We investigate the performance of *TCTC*(*N*, *K*) in terms of switching throughput. The switching throughput ρ IS defined the ratio of the average number of cells arrived at output ports over the average number of cells arrived at input ports. In order to simply the analytical work, we assume that each input in *IM* or *OM* has a buffer arranged as FIFO queue for arriving cells. No output buffer in *IMs*, i.e. a cell being switched through the fabric of *IM* is forwarded to its corresponding *OM* and is buffered in its input buffer. The arriving traffic is Bernoulli i.i.d. uniform pattern.

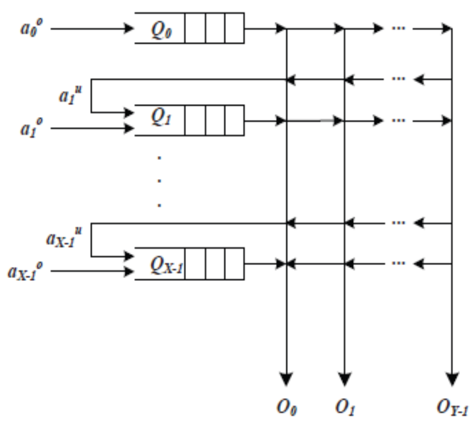

Fig. 3. Queueing network model of *CTC*(*X*, *Y*)

A. Queueing Model of CTC(X, Y)

An *IM* (*OM*) is a $k \times m$ ($m \times k$) contention tolerant crossbar, where $m = N/k$. Therefore, let's consider a general contention-tolerant crossbar model with *X* inputs and *Y* outputs, denoted as *CTC*(*X*, *Y*). We model a *CTC*(*X*, *Y*) as a semi-open queueing network system, as shown in Fig. 3.

Each input buffer (i.e. queue) is organized as an FIFO queue, denoted by Q_i. The Head-of-line cell (if exist) of an input queue will be transmitted to corresponding output line within one time slot. where α_i^o and α_i^u and the arrival rate of Q_i from outside of $CTC(X, Y)$ and upstream input of Q_i, respectively. Let θ_i be the average transmission rate of cells passing through Q_i. $p_{k,j,i}$ is the probability of a cell leaving Q_k for Q_i by output line *j*. We consider two possible cases:

Case I: Q_i is under unsteady state, i.e. $\alpha_i^o + \alpha_i^u \geq 1$.

Since at most one cell can be transmitted to output for each input without speedup, the saturated transmission rate is 1. We have $\theta_i = 1$.

Case II: Q_i is under steady state, i.e. $\alpha_i^o + \alpha_i^u < 1$.

The traffic equation which is held for Q_i is

$$\theta_i = \begin{cases} \alpha_i^o & if \quad i = 0 \\ \alpha_i^o + \sum_{j=0}^{Y-1}\left(\sum_{k=0}^{i-1} p_{k,j,i}\theta_k\right) & if \quad 0 < i \leq X-1. \end{cases} \tag{1}$$

From the property of $CTC(N)$, one cell leaves Q_k for its downstream Q_i if and only if they both transmit their cells to the same output column at the same time slot. Thus we have

$$p_{k,j,i} = \begin{cases} p_{k,j}(p_{i,j}\theta_i) & if\ k = i-1; \\ p_{k,j}\left[\prod_{m=k+1}^{i-1}(1 - p_{m,j}\theta_m)\right] p_{i,j}\theta_i & if\ 0 \leq k < i-1. \end{cases} \tag{2}$$

where $p_{i,j}$ is the probability of a cell being chosen to transmit to O_j from Q_i. For uniform traffic, $p_{i,j} = 1/Y$. Combining (2) and solving (1) iteratively, we obtain (3)

$$\theta_i = \frac{Y\alpha_i^o}{Y - i\alpha_i^o} \tag{3}$$

Let γ_j be the average rate of cells achieving O_j. Concluding above two cases, we have equations (4).

$$\gamma_j = 1 - \prod_{i=0}^{X-1}(1 - p_{i,j}\theta_i) \tag{4}$$

where

$$\theta = \begin{cases} \dfrac{Y\alpha_i^o}{Y - i\alpha_i^o}. & if\ 0 \leq i < (\frac{1}{\alpha_i^o} - 1)Y. \\ 1 & if\ (\frac{1}{\alpha_i^o} - 1)Y \leq i \leq X-1. \end{cases} \tag{5}$$

CTC(*N*) can be seen as *CTC*(*N*,*N*), the simulations in [7] prove the correctness of theoretical results.

B. Throughput Analysis of TCTC(N, k)

In order to identify the different parameter of *IM* and *OM* in *TCTC*(*N*, *k*), let $I_i^{IM_d}$ be the i^{th} input of IM_d; $O_j^{IM_d}$ be the j^{th} output of IM_d; $I_i^{OM_d}$ be the i^{th} input of OM_d; $O_j^{OM_d}$ be the j^{th} output of OM_d. Under the uniform traffic assumptions, all inputs of *IMs* have the same offered load, and outputs of IMs have the same throughput. Since $O_j^{IM_d}$ connects to $I_i^{OM_d}$,the traffic pattern of *OMs* are also Bernoulli i.i.d uniform. Therefore, let α^o be the traffic arriving rate of (offered load) of *IMs*; γ be the rate of cells achieving $O_j^{IM_d}$, $0 \leq j \leq \frac{N}{k} - 1, 0 \leq d \leq \frac{N}{k} - 1$; $\hat{\alpha}^o$ be the traffic arriving rate of OMs; $\hat{\gamma}$ be the rate of cells achieving $O_{\hat{j}}^{OM_{\hat{d}}}$, $0 \leq \hat{j} \leq \frac{N}{k} - 1$, $0 \leq \hat{d} \leq \frac{N}{k} - 1$. Obviously, we have $\gamma = \hat{\alpha}^o$. Meanwhile, the i^{th} input of have the $IM_d(OM_{\hat{d}})$ have the same transmission rate. Therefore, we use θ_i to represent the transmission rate of i^{th} input of IMd, and $\hat{\theta}_i$ to represent the transmission rate of *i*th input of $OM_{\hat{d}}$. Let λ be the offered load of *TCTC*(*N*,*k*), thus we have $\alpha^o = \lambda$.

An *IM* is a $CTC(k, \frac{N}{k})$ and an OM is a $CTC(\frac{N}{k}, k)$. From the solutions in section III-A, for IM_d, $0 \leq d \leq \frac{N}{k} - 1$, and $OM_{\hat{d}}$, $0 \leq \hat{d} \leq \frac{N}{k} - 1$,we have

$$\theta_i = \begin{cases} \dfrac{\frac{N}{k}\lambda}{\frac{N}{k} - i\lambda}, & \text{if } 0 \leq i < \left(\dfrac{1}{\lambda} - 1\right)\dfrac{N}{k} \\ 1, & \text{if } \left(\dfrac{1}{\lambda} - 1\right)\dfrac{N}{k} \leq i \leq k-1 \end{cases} \tag{6}$$

$$\gamma_j = 1 - \prod_{i=0}^{k-1}(1 - \tfrac{k}{N}\theta_i) \tag{7}$$

$$\hat{\theta}_i = \begin{cases} \dfrac{k\gamma}{k - i\gamma}. & if \quad 0 \leq i < (\frac{1}{\gamma} - 1)k \\ 1 & if \quad (\frac{1}{\gamma} - 1)k \leq i \leq \frac{N}{k} - 1 \end{cases} \tag{8}$$

and

$$\hat{\gamma}_j = 1 - \prod_{i=0}^{\frac{N}{k}-1}\left(1 - \frac{1}{k}\hat{\theta}_i\right) \tag{9}$$

According to the definition of switch throughput, we obtain:

$$\rho = \frac{N \times \hat{\gamma}}{N \times \alpha^o} = \frac{\hat{\gamma}}{\lambda} \tag{10}$$

Combining (4)-(7), and (8), the switch throughput of *TCTC*(*N*, *k*) can be computed.

Fig. 4 shows the throughput comparison of *CTC* with switch size 64, 128, and 1024, and corresponding *TCTCs* with *k* having value 2 and 8. From the results, we can see that *TCTC*(*N*, *k*)s achieve higher throughput with smaller *k*. *TCTC*(1024, 2) nearly achieves 100% throughput.

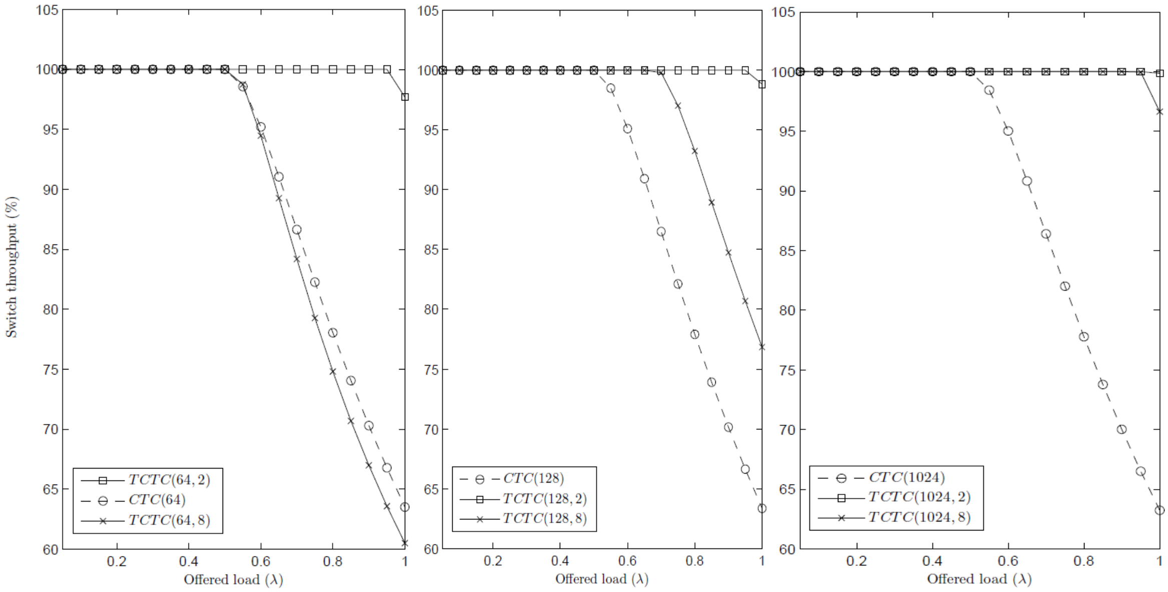

Fig. 4. Throughput comparison of CTCs and TCTCs.

V. Conclusion

As it was proved in [7], the throughput of *CTC*(*N*) with FIFO input queues is bounded by 63%. In this paper, we presented a new architecture using small *CTC* components called Two-Stage Contention-Tolerant Crossbar, denoted as *TCTC*(*N*, *k*), And we proved that it achieves high throughput by developing the its queuing model.

References

[1] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, “Achieving 100% throughput in an input-queued switch,” *IEEE Trans. on Commun.*, vol. 47, no. 8, Aug. 1999.

[2] A. Gourgy, T. H. Szymanski, and D. G. Down, “On tracking the behavior of an output-queued switch using an input-queued switch,” *IEEE Trans. on Networking*, vol. 17, no. 6, Dec. 2009.

[3] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, “High speed switch scheduling for local area networks,” *ACM Trans. on Computer Systems*, vol. 11, no. 4, pp. 319-352, Nov. 1993.

[4] N. McKeown, “The iSLIP scheduling algorithm for input-queued switches,” *IEEE/ACM Trans. on Networking*, vol. 7, no. 2, pp. 188-201, Apr. 1999.

[5] R. Rojas-Cessa, E. Oki, and H. J.Chao, “CIXOB-k: Combined input- and crosspoint-queued switch,” *IEICE Transactions on Communications*, vol. E83-B, no. 3, pp. 737-741, Mar. 2000.

[6] R. Rojas-Cessa, E. Oki, and H. J. Chao, “On the combined input-crosspoint buffered switch with round robin arbitration,” *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1945-1951, Nov. 2005.

[7] S. T. Chuang, S. Iyer, and N. McKeown, “Practical algorithms for performance guarantees in buffered crossbars,” *INFOCOM 2005*, vol. 2, pp. 981- 991. 13-17, Mar. 2005.

[8] N. Chrysos and M. Katevenis, “Crossbars with minimally-sized crosspoint buffers,” in *Proc. IEEE Workshop on High Performance Switching and Routing*, June 2007.

[9] S. M. He, S. T. Sun, H. T. Guan, Q. Zhang, Y. J. Zhao, and W. Gao, “On guaranteed smooth switching for buffered crossbar switches,” *IEEE/ACM Trans. on Networking*, vol. 16, no. 3, pp. 718-731, 2008.

[10] G. Qu, H. J. Chang, J. Wang, Z. Fang, and S. Q. Zheng, “Contention-tolerant crossbar packet switches,” *International Journal of Communication Systems*, vol. 24, pp. 168-184, 2011.

[11] G. Qu, H. J. Chang, J. Wang, Z. Fang, and S. Q. Zheng, “Designing fully distributed scheduling algorithms for contention-tolerant crossbar switches,” in *Proc. Internetion Conference on High Performance Switching and Routing*, Dallas, 2010.

[12] G. Qu, H. J. Chang, J. Wang, Z. Fang, and S. Q. Zheng, “Queueing analysis of multi-layer contention-tolerant crossbar switch,” *IEEE Communications Letters*, vol. 14, no. 10, Oct. 2010.

[13] G. Qu, Z. Fang, H. J. Chang, X. Zhao, and S. Q. Zheng, “DiaCTC(N)-An improved contention- tolerant crossbar switch,” *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 53-65, 2013.

[14] Z. Fang, L. Xiang, G. Qu, J. Zhang, and S. Q. Zheng, “Parallelized contention tolerant crossbar packet switch,” *Journal of Computational Information Systems*, vol. 10, no. 19, pp. 8169-8176, 2014.

[15] J. Zhang, Z. Fang, G. Qu, X. Zhao, and S. Q. Zheng, “On out-of-sequence problem in contention-tolerant crossbar switches,” *Information Technology Journal*, vol. 10, no. 12, pp. 2268-2275, 2011.

[16] G. Qu, Z. Fang, J. Zhang, and X. Zhao, “An analytical model of contention-tolerant crossbar switch,” *Journal of Computational Information System*, vol. 8, no. 17, pp. 7045-7052, 2012.

Zhiyi Fang received the Ph.D. degree in computer science from Jilin University, Changchun, China, in 1998, where he is currently a professor of computer science. He was a senior visiting scholar of the University of Queensland, Australia, from 1995 to 1996, and the University of California, Santa Barbara, from 2000 to 2001. He is a member of China Software Industry Association (CSIA) and a member of Open System Committee of China Computer

Federation (CCF). His research interests include distributed/parallel computing system, mobile communication, and wireless network.

Shuaibing Lu received the M.S. degree in computer science and technology from Jilin University, Changchun, China, and she is currently a Ph.D. Student. She was a researcher at the distributed computing system and network access control algorithm and vehicle information systems. She has published a national patent.

Guannan Qu received the B.E. degree in computer science and technology from Jinan University, Guangzhou, China, in 2003, and the M.S. and Ph.D. degrees in computer science from Jilin University, Changchun, China, in 2007 and 2010, respectively. She is currently a lecturer of computer science and technology in Jilin University. She was financially supported by China Scholarship Council as a visiting scholar in the University of Texas at Dallas, from 2008 to 2010. Her research interests include the areas of computer system architecture, performance analysis, and quality-of-service issues in high-speed networks. She is a member of the IEEE.

Jianfei Zhang received the B.E. degree, M.S. degree and Ph.D. degree in computer science and technology from Jilin University, Changchun China, in 2004, 2007 and 2013, respectively. Now he is a teacher and research fellow of computer science and technology in Changchun University of Science and Technology, China. He was financially supported by China Scholarship Council as a visiting scholar in the University of Texas at Dallas, USA, from 2010 to 2013. His research interests are in the areas of computer system architecture, interconnection networks and data center.

Si-Qing Zheng received the Ph.D. degree from the University of California, Santa Barbara, in 1987. After being on the faculty of Louisiana State University for eleven years, he joined the University of Texas at Dallas in 1998, where he is currently a professor of computer science, computer engineering, and telecommunications engineering. Dr. Zheng's research interests include algorithms, computer architectures, networks, parallel and distributed processing, telecommunications, and VLSI design. He has published in these areas extensively. He was a consultant of several high-tech companies, and he holds numerous patents. He served as program committee chairman of numerous international conferences and editor of several professional journals.