Received: 30 September 2020 | Revised: 2 April 2021 | Accepted: 12 May 2021

IET Communications

DOI: 10.1049/cmu2.12236

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH PAPER

CQPPS: A scalable multi-path switch fabric without back pressure

Boyan Pan | **Qingqing Zhou** | **Guo Chen**

Department of College of Computer Science and Electronic Engineering, Hunan University, Changsha, China

Correspondence
Qingqing Zhou, Department of College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China.
Email: qqzhou@hnu.edu.cn

Boyan Pan and Qingqing Zhou are co-primary authors.

Funding information
Fundamental Research Funds for the Central Universities, Grant/Award Number: 531118010153; Training Program for Excellent Young Innovators of Changsha, Grant/Award Number: Z202069420755; National Natural Science Foundation of China, Grant/Award Number: 61872132

Abstract

Generally, in order to guarantee a good throughput and decrease the complexity of reassembling, the majority of current commodity routers take elaborate closed-loop flow control schemes such as back pressure to prevent cell loss in the switch fabrics. As commodity router's port number grows larger and link rate becomes faster, the huge I/O pins and memory consumption makes these closed-loop flow controls very difficult to implement engineeringly. This paper approaches the problem of building ultra-large-capacity router from a different angle. Crosspoint-queued-based Parallel Packet Switch (CQPPS), a highly scalable switch architecture with no need of any closed-loop flow control schemes, is proposed. And the authors propose the Padded Frame plus Round-Robin scheduling scheme for CQPPS architecture. By allowing potential cell loss in the switch fabrics and slightly higher light-load delay, CQPPS achieves loss rate orders of magnitudes lower than back pressure schemes, and high-load delay 10 times less than back pressure schemes. It also greatly reduces the complexity of engineering implementation.

1 | INTRODUCTION

As content-rich Internet applications such as video streaming, audio streaming and file sharing become more and more popular, the demand for higher backbone bandwidth has grown extremely fast. Therefore, designing high-speed switching fabric, a critical building block for backbone routers, has become more and more important. Limited by the single-chip capacity, traditional single-path switch fabrics are no longer capable to meet the demand of modern network. Thus both industry and academia focus on multi-path switch fabrics that consist of multiple switching chips [1–4].

Typically, in convenience of clock synchronization and memory access, routers fragment packets into fixed-length cells and then transmit them into switch fabrics. After cells are switched to the destination ports, they are reassembled into original packets at the output line cards. Generally, in order to guarantee a good throughput and decrease the complexity of reassembling, current commodity routers (such as Cisco CRS carrier routing system 16-slot line card chassis enhanced router system [1]) have to take elaborate closed-loop flow control schemes to prevent cell loss in the switch fabric, namely, back pressure. Back pressure is often implemented in three ways: (1) transmitting congestion signals through reserved wires, (2) sending control message packed in special cells using data paths (we divide the credit based methods [5] into this category) and (3) a combination of the former two methods. Back pressure works well for switches with low-speed and small number of ports, but has serious limitations (which will be detailed in Section 2): (1) engineering complexity; (2) large number I/O pin consumption; (3) large memory consumption; (4) saturation tree-caused performance degradation and (5) communication overhead.

Realizing back pressure's scalability limitations, research communities have proposed various improvement on closed-loop flow control schemes, such as [3, 6–8]. However, these schemes still suffer the engineering complexity issues and some might even suffer worse scalability problems under certain conditions. Thus, none of the approaches have been adopted by commodity routers.

This paper approaches the problem of building ultra-large-capacity router from a different angle : instead of trying to improve back pressure scheme, can we get rid of back pressure or closed-loop flow control in the switching fabric all together? Back pressure or other closed-loop flow control schemes all try to guarantee there is no cell loss in switch fabric. What needs to be explained here is that 'no cell loss in switch fabric' does

not mean there will be no packet loss rate in the back pressure schemes. Although traditional back pressure schemes can prevent cell loss in switch fabrics, it cannot prevent packet loss because the back pressure signals returned due to conflicts can cause packets to be discarded at the entrance. Considering this, can we just loosen the requirement that no cell can be lost in the switch fabrics to achieve better overall router performance?

Along above direction, we proposed Crosspoint-queued-based Parallel Packet Switch (CQPPS), a highly scalable switch fabric with no need of any closed-loop flow control schemes. Our contributions can be summarized as follows:

- We proposed CQPPS, the first switch fabric architecture without the need of back pressure or any closed-loop flow control.
- Based on the CQPPS architecture, we present a simple scheduling scheme named Padded Frame plus Round-Robin (PFRR) which achieves good performance without back pressure or any closed-loop flow control.

By allowing potential cell loss in the switch fabrics and slightly higher light-load delay, CQPPS along with PFRR achieve loss rate orders of magnitudes lower than back pressure schemes, and high-load delay 10 times less than back pressure schemes.

The remainder of the paper is organized as follows. Section 2 details the limitations of back pressure. Section 3 overviews CQPPS architecture. Section 4 presents PFRR scheduling scheme. Section 5 uses simulations to compare performance between CQPPS and other schemes. Section 6 discusses the implementation issues and some future work. Finally, Section 7 concludes the paper.

2 | BACK PRESSURE: LIMITATIONS AND PROBLEMS

This section details the limitations and problems faced by back pressure.

Besides the well-known complexity of engineering implementation, back pressure scheme has the following limitations as the link speed and the port number grows.

I/O pin consumption:
To avoid the HOL (Head-of-Line) problem, an easy way to implement back pressure in small-scale switch fabrics is to allocate physical wires for every flow. Back pressure signals use these reserved wires to notify the upper stage of congestion. As the demands of switch capacity grows fast, modern core routers (such as Cisco crs carrier routing system 16-slot line card chassis enhanced router system) mostly use multi-path switch architecture which consists of multiple low-speed switching planes [2]. As the switching paths become more and more, physical resource separation could consume a large number of I/O pins of switching chips. Assuming we implement an $N \times N$ switch fabric using a multi-plane switching fabric with K central planes. In order to avoid the HOL problem, each of the N router outputs needs a unique back pressure signal to each of the N router inputs, which resulting in N^2 back pressure I/O pins on each switching plane. Besides, each of the N demultiplexer outputs needs a unique back pressure signal to each of the K central plane inputs, then $N \times K$ I/O pins are need to be reserved between N demultiplexers and K central planes (one per output per switching plane). Assuming we use eight switch planes, each of the switch planes will work at 1/8 of the external link rate so that total of them can match the external input link rate. Figure 1 shows the I/O pins requirement of back pressure as the switch port number grows for multi-plane switch with each plane working at 1/8 of the external link rate. When we looked at the latest FPGA chips (Such as Intel Stratix 10 TX 850 FPGA chips, Intel Stratix 10 GX FPGA etc.) [9] for comparison, the requirement of I/O pins has exceeded the number of user I/O pins (available pins that is programmable for users) while the port number grows more than 64. Using ASIC chips with more available pins than FPGAs still cannot remove the bound shown in the figure, but only improve it. This becomes a big hurdle to build a switch fabric with large scale.

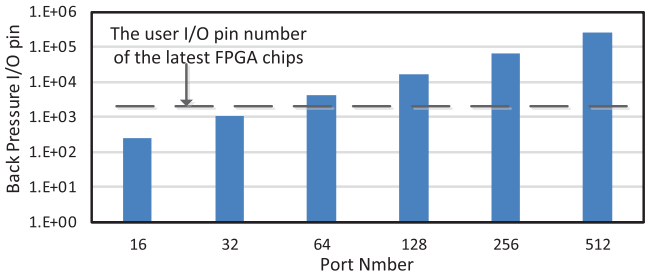


FIGURE 1 Multiplane switch with eight switch plane each working at 1/8 of the external link rate: I/O pins consumption of back pressure as the port number grows

Memory consumption:
Furthermore, the modern traditional central switch modules (SMs) are OQ (Output-Queued) switches. When considering the delay of back pressure feedback, switch chips have to allocate large buffers to prevent cell loss. With growth of the link speed, delay for back pressure signals transporting and processing becomes larger and larger compared to data's transporting. The buffer in the receiver module should be large enough to hold all the packets on the fly during this time period. As [10] shows, in order to reduce power consumption, line cards and SM in modern core routers are often placed in different racks with distance up to tens of meters. Now, let us take the 60 m which is a more moderate number for example. Assuming the signal propagation speed is 1×10^8 m/s [10], the back-of-envelope calculation shows that the feedback delay of back pressure signals can be longer than 600 ns even without considering the processing delay. Taking a 64×64 switch fabric of eight switch planes with no speedup as an example. We need to calculate the buffer size in the receiver module for each switch plane, so we need to first calculate the link rate of each switch plane, which is $1/N$ of the external link rate. For a 100-Gbps input link, the link rate of each switch plane is 12.5 Gbps. And each switch plane has 64×64 back pressure signals, so the buffer size

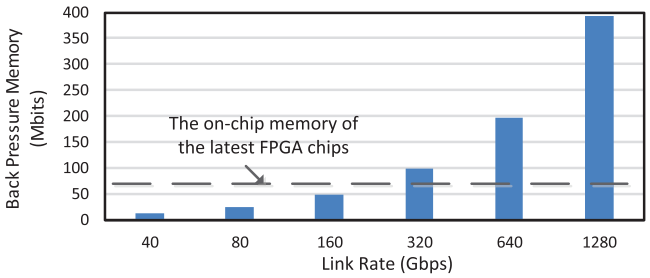


FIGURE 2 64 × 64 switch fabric with eight switch plane each working at 1/8 of the external link rate: Memory consumption of back pressure as the link rate grows

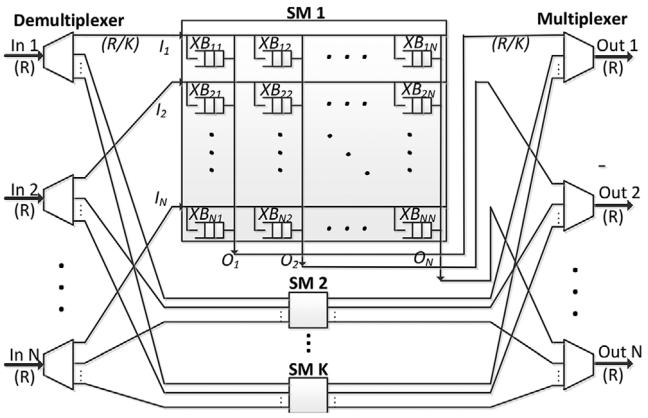


FIGURE 3 The CQPPS switch fabric

should be multiply by 256. Therefore, a buffer in the receiving stage should be at least 30.7K bits in guarantee of no buffer overflow. As the link with such speed, all the buffers have to be implemented using on-chip memories, because off-chip DRAM (Dynamic Random Access Memory) and SRAM (Static Random Access Memory) are not fast enough for accessing. Figure 2 shows, taking a 64 × 64 switch fabric of eight switch planes with no speedup as an example, how memory consumption increases as the link rate speed grows faster. For switch fabrics of such scale and of link rate higher than 320 Gbps, the memory requirement exceeds the Intel Stratix 10 TX 850 FPGA (Field Programmable Gate Array) chips' capacity (about 60 Mbits) [9]. Similar to I/O pins, using ASIC (Application Specific Integrated Circuit) chips cannot remove the limitation either. Thus, the demand of a buffer's capacity also limits the scale of a high speed switch fabric.

Saturation tree:

The saturated tree problem [11] occurs in the multi-level network of distributed routing. It is easy to have hot spot contention in the network. When some output ports send too much messages to a target input port, the target port becomes a hot spot. The messages fed back by the target port will fill the buffer of the network nodes behind the target port, forming a root saturated tree [12]. As the resource requirement of back pressure dramatically increases as the growing of the switch fabrics' scale, methods that integrate multiple congestion information into few back pressure signals came up. However, the back pressure scheme can easily cause HOL blocking if multiple flows share the same back pressure signal. And this can lead to a saturation tree problem on the network. Back pressure signals from a hot spot will block all the paths destined to it which forms a saturation tree. Any other flows which pass these links wait unnecessarily until the back pressure signals from the hot spot are lifted. As the port number of a switch fabric grows, HOL blocking is more likely to occur, which dramatically weaken the switching performance. Our simulations in Section 5 will show how saturation tree problems degrade the performance.

Communication overhead:

Another direction to save I/O pins is to transmit back pressure signals through data paths. However, this will lead to large communication overhead caused by the control messages. When the links are under heavy load, it becomes very difficult to handle the control messages transmitting through congested data paths.

3 | CQPPS OVERVIEW

As Figure 3 shows, CQPPS consists of three parts: demultiplexers, SMs and multiplexers. SM is like a crossbar which is composed of $N \times N$ crosspoints, and each of the crosspoints has a crosspoint queue. For an $N \times N$ CQPPS switch, assuming that each input and output link rate is R and there are K central SMs, then K different switching paths exist between a pair of input and output. Each internal link rate between SMs and demultiplexers or multiplexers is R/K. Apparently we can see that, a switch fabric with high-speed links and large number of ports could be built using several low-speed SMs with the same number of ports in CQPPS architecture. Note that we are presenting a switching solution for fixed-length cell switching and reassembling. So we consider that time is slotted and packets have already been chopped into fixed-length cells before being sent into the demultiplexer. A flow in the switch fabric is defined as the packets coming from the same input and going to the same output. We follow the above assumptions in the rest of this paper.

Instead of the original OQ switches, we use CQ (Crosspoint-Queued) switches as the central SMs to build a PPS (Parallel Packet Switch) fabric [13]. All the CQ SMs are of the same structure. Flows are stored in different crosspoint queues separately. Although our structure uses crosspoint crossbar buffering results in better performance, it requires more on-chip memory. This caused some difficulties with the storage implementation. But with the latest FPGA chip and technology, it was more than enough to implement [14]. And we will show later that this structural modification is the fundamental step to get rid of complex closed-loop flow control schemes.

We briefly introduce our scheduling scheme here. We adopt UFS (Uniform Frame Spreading) method [15] in demultiplexers to dispatch cells of the same flow simultaneously to all the central SMs. We ensure the links between the demultiplexer and each SM are with the exactly same physical length and rate.

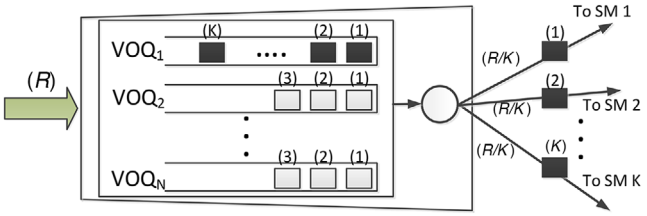


FIGURE 4 CQPPS demultiplexer

Thus, all the cells dispatched in the same time will arrive at all the SMs around the same moment. Also, we have all the SMs driven by the same clock in the switch board. Then, with the same scheduling algorithm in each SMs, all the central SMs will behave the same. Section 6 discusses in more details how to make sure that a slight arrival time difference will not affect SMs' behaviour. By making all central SMs behave the same, our methods could reach a good performance with no need of closed-loop flow control and could also naturally maintain cells sequence.

In the next section, we will introduce in details that, without closed-loop flow control, how our schedule scheme PFRR works in the CQPPS switch fabric.

4 | PFRR SCHEDULE SCHEME

Getting rid of complex closed-loop control schemes means that cells may be lost in the middle switch paths. Switch fabrics in the past do not allow cells lost in middle SMs because (1) cell loss may lead to a very low throughput since that many packets may be not integral because of the miss of just several cells, and (2) the output reassembling would be very difficult to judge whether or not a cell is lost in the middle switching modules. PFRR schedule scheme has to solve the two problems brought by cell loss stated above. Furthermore, a good schedule scheme for multi-path switch fabrics should also take the following two factors into consideration: (1) Good load-balance to uniformly distribute cells into all switch paths and (2) limiting cells OoS (Out-of-Sequence) which greatly infects the difficulty of reassembling. PFRR satisfy well these demands of multi-path switch fabrics.

Next, we introduce the scheduling scheme in details.

4.1 | Demultiplexer

As Figure 4 shows, there are N VOQs (Virtual Output Queues) storing cells destined to N different outputs. We adopt an algorithm based on the UFS method and the PF (Padded Frame) method [16] to maintain the cells' sequence in the same flow. We used a clock to pick a VOQ. Once a VOQ has buffered more than K cells, we dispatched the first K cells simultaneously to K SMs. We assume the links between the demultiplexer and each SM are with the exactly same length and rate. Thus, all the cells dispatched simultaneously will arrive at the SMs at the same time. What needs to be explained here is that the 'simultaneously' and 'at the same time' are not a rigorous statement. We divided every clock into k slots, and these k cells will be sent out in order in each slot. In every clock, they look like they are being sent at the same time. We will discuss more about the clock frequency in the discussion section. Therefore, in order to simplify the description in the following, sometimes we will say that k cells are sent 'simultaneously' or 'at the same time'. All K cells dispatched at the same time are called in the same 'round'. We use PF methods to reduce the dispatching delay in multiplexers. However, the schemes we adopted to pad the frame are totally different from [16] and do not need any communication between demultiplexers and SMs. The dispatching and padding action only happen at the end of every K time slots. If there are no VOQ's length exceeding K at this moment, we use 'fake' cells to pad the queue with the oldest head cell, into K cells and dispatch them out. The scheduling procedure in the demultiplexer is shown in Algorithm 1.

ALGORITHM 1 Demultiplexer scheduling algorithm

Initialize:

wTime stores the number of time slots since the VOQ received the first cell after each time *wTime* has been reset

During each time slot:

1: **if** $wTime \geq K - 1$ **then**
2: pick up the longest VOQ as *LongestQueue*
3: **if** *LongestQueue.length* $\geq K$ **then**
4: $wTime = 0$
5: dispatch the first 1 ... K cells in VOQ *LongestQueue* to path 1 ... K,respectively, at the same time
6: **else if** at least one VOQ is not empty **then**
7: $wTime = 0$
8: pick up the VOQ *OldestQueue* with the oldest head cell and pad theVOQ with $K -$ *OldestQueue.length* 'fake' cells
9: dispatch the first 1 ... K cells in VOQ *OldestQueue* to path 1 ... K,respectively, at the same time
10: **end if**
11: **else if** *wTime* 0 **then**
12: $wTime = wTime + 1$
13: **else if** $wTime = 0$ **then**
14: **if** at least one VOQ is not empty **then**
15: $wTime = wTime + 1$
16: **end if**
17: **end if**

Line 1 of Algorithm 1 ensure that no more than K cells (including 'fake' cells) will be dispatched in K time slots, thus

the links between demultiplexers and SMs are ensured to be not oversubscribed. If there is no VOQ having more than K cells and there is at least one VOQ is not empty and K time slots have passed since the demultiplexer dispatched cells last time, it selects the VOQ with the oldest head cell and pad it into K cells and send out them.

Then we prove that the VOQ's length has an exact bound which is $(N-1)\times K$. We analyse one demultiplexer with no loss of generality. For ease of clarification, we assume that cells arrive before the dispatching process in each time slot. It is obvious that there are at most $(N-1)\times K$ cells buffered in all VOQs. We assume that in the first K time slots, $K-1$ cells are buffered in the first VOQ, and the remaining 1 cell is buffered in the last VOQ. In the second K time slots, $K-1$ cells are buffered in the second VOQ, and the remaining 1 cell is buffered in the last VOQ. By analogy, until the $N-1$th time slots is completed, the previous $N-1$ VOQs each buffer $K-1$ cells, and the last VOQ buffers $N-1$ cells. So a total of all VOQs $(N-1)\times K$ cells are buffered. And the total number of cells buffered in all VOQs will no longer increase. We call this state the worst case. After the worst case, the VOQ's length will not grow. So the bound of VOQ's length is $(N-1)\times K$. It is easy to calculate the exact bound using numerical methods given the value of N and K. Besides, when this is at least one VOQ has more than K cells, we select the longest VOQ to send out in our algorithm. So it is no necessary to worry about the worst case.

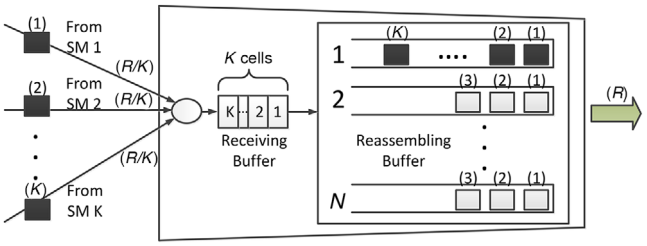


FIGURE 5 CQPPS multiplexer

ALGORITHM 2 Reassembling algorithm

For a reassembling buffer:

```
if the head cell is not the SOP (Start Of a Packet) then
    drop the head cell
else
    find the nearest EOP (End of a Packet) cell
    if there is a EOP cell then
        reassemble those cells between SOP and EOP (including them)into a packet and send it out
    else
        wait
    end if
end if
```

4.2 | Switch module

As illustrated in SM 1 of Figure 3, CQ switches are used as SMs in CQPPS. Same work-conserving Round-Robin algorithms are used in all these SMs. We define XB_{ij} that the crosspoint buffer for cells from I_i destined to O_j.

The scheduling process of a CQ switch can be described as the following two steps:

- **Arrival Step**:If there is a cell arriving at input i heading to the output j at the start of a time slot, it is buffered in XB_{ij} in a FIFO (First Input First Output) manner if the buffer is not full. The cell will be dropped in the case of that XB_{ij} is full.
- **Departure Step**:Within the same slot, the scheduler of each output independently selects one XB in its column according to the Round-Robin pointer, and sends the HOL cells out of the switch through the output. If the selected XB is empty, select the nearest non-empty XB to the pointer and send the HOL cell. If all XBs in this column are empty, no cell will be transmitted by this output in this time slot. Note that the departure steps at different output schedulers are run in parallel. After that, make the pointer point to the next XB.

As mentioned before, K cells in a round, which are of a same flow, will arrive at all SMs simultaneously. Because they all use the same Round-Robin scheduling algorithms depicted as above, all SMs will behave exactly the same. Then, these K cells will be switched to the output multiplexer at the same time or dropped in each SM if the corresponding buffers are full. By doing this, cells will be dropped in the unit of K. This is an important step to guarantee the throughput.

4.3 | Multiplexer

Figure 5 shows the structure of multiplexers in CQPPS. Each multiplexer contains one receiving buffer and N reassembling buffers to reassemble cells of N flows. Because all SMs use the same clock, then K cells (including 'fake' cells) of a round will arrive simultaneously. Cells in the receiving buffer are buffered in the position according to their coming paths in the same manner as dispatched by the demultiplexer. So cells are naturally in-sequence in the receiving buffer. If the receiving buffer collects K cells, it transmits them all in-sequence to the reassembling buffer of their flow. 'Fake' cells are dropped in the reassembling buffer and true cells are reassembled to original packets. Since there is no closed-loop flow control, cells may be dropped in the SMs. So the reassembling algorithm has to deal with this situation. Algorithm 2 shows that how cells are reassembled. In the cells of a packet, there are two special cells (the start-of-packet [SOP] cells and the end-of-packet [EOP] cells) that mark the start and end of the packet. The SOP and EOP cells are generated when the packet is cut at the entrance. The reassembling buffer simply detects SOP and EOP cells. The cells between SOP and EOP are regarded as an integral packet and reassembled. It is unnecessary to worry that the length of

the packet will greater than K. Because K is only a threshold, and the reassembling buffer size is greater than K. The size of the reassembling buffer only needs to be greater than 1500 bytes (the length of the largest Ethernet packet). If there are some cells in a packet lost but SOP and EOP cells have been successfully switched to the multiplexer, a bad packet will be produced. It will be detected having the wrong checksum and dropped by the next stage to the switch fabric. This will not affect the follow-up reassembling procedure. In addition, very few cells will be dropped as shown in simulations in Section 5. If there are non-SOP cells appeared in the head of reassembling buffer, they are dropped immediately. Because this means that the original SOP cell has been lost in SMs.

5 | REAL-TRACE SIMULATION

5.1 | Experiment setup

In this section, we conduct several simulations using real traces. Our data consists of two parts from CAIDA (Cooperative Association for Internet Data Analysis), two 1-min traces from 10 Gbps links, one at San Jose and another from Chicago. We compare four multi-path switch scheduling methods with original back pressure flow control schemes with our method. These methods can preserve the flow sequence while offer relative good performance. We do not consider those scheduling algorithms which require a lot of instant communication between different stages such as [17] and [16].

Among the schemes we simulated, the first is static hash (SH) methods, which chooses the switch path of each packet according to the result of XOR-hashing the 5-tuple. The second is adaptive table hash (ATH) methods [18] with 320 table entries. ATH adjust the switching path allocation every 1000 time slots in a Round-Robin manner. The third method is flow slice (FS) [19], with the slicing threshold set to be 0.01 ms. And the last method is Round-Robin with Virtual Input Queue (RRVIQ) [20]. The first three are called flow-based methods.

We suppose to simulate a 16×16 switch fabric with 100 Gbps link rate using given chips. Because the link rate is very fast, all buffers are implemented by on-chip memory considering the memory access speed. We consider that our model are to be implemented using the same latest FPGAs, which has approximately 60 Mbits [9] on-chip memory. And the size of our cells is 64 bytes (the size of the smallest Ethernet packet). Therefore, we can compare all different methods fairly under the same hardware resources restriction.

We divide a 60-s trace into 16 equal size segments and compress them by different times to emulate different input loads for all inputs. The destination port of each packet is set as the destination IP address modulo 16. The traffic distribution under this situation is shown in Figure 6. As we can see, the traffic for simulation is not uniform. We believe this reflects the real condition of the Internet. Note that the input traffic maybe not strictly admissible because of compression. Approximately 2.1×10^8 packets with total length of 1.72×10^{11} bytes are sent into the switch fabric during each experimentation. In the rest of the paper, we only present the simulation result of Chicago trace due to space limitation. The results for San Jose trace are similar.

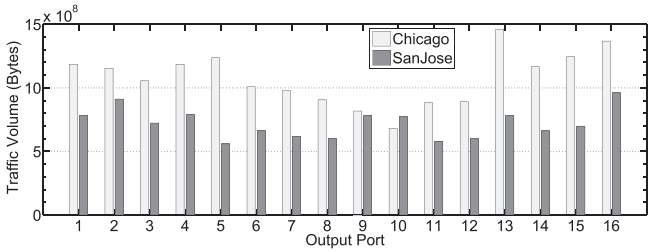


FIGURE 6 Traffic distribution to each output

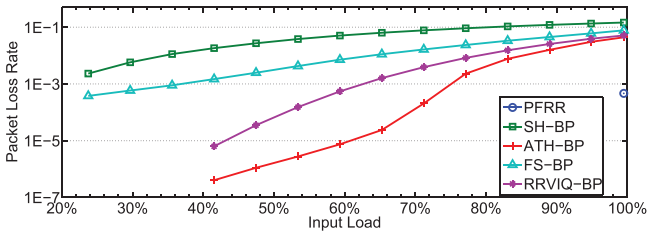
We conduct comparisons under three different implementations of those methods: (1) in Parallel Packet Switch consists of Output-Queued SMs (OQPPS) [21] architecture with physical-wire back pressure, (2) in OQPPS architecture with data-plane back pressure and (3) and in Clos architecture with physical-wire back pressure scheme.

5.2 | Results

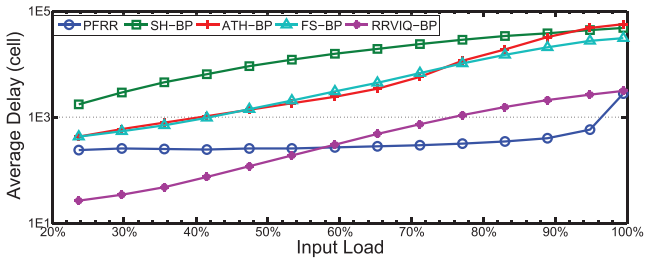
5.2.1 | Comparison with schemes with physical wire back pressure

First, we compare the packet loss rate and average delay of our method PFRR in CQPPS, to all methods in OQPPS architecture implemented with separate physical-wire back pressure (in the later figures, it is going to be called BP for short) signals. Congestion signals are feedback through extra physical wires, thus will not occupy the data channel. The OQPPS has the same structure as the CQPPS shown in Figure 3, but consists of OQ switch as SMs. We set eight SMs in the middle and each works at 12.5 Gbps transmission rate, to build a whole 16×16 switch fabric with 100 Gbps link rate. Then in every demultiplexer, SH, ATH, FS and RRVIQ method all need 8×16 VOQs for each input to avoid the HOL problem caused by back pressure. For each SM, PFRR needs 16^2 crosspoint queues and the other four methods need 16 output queues. Assuming every demultiplexer, multiplexer and SM chip has the same 60M bits on-chip memory, then each VOQ's capacity in demultiplexers of the four methods is 469 Kb and each OQ is 3.75 Mb. And for PFRR in CQPPS, each crosspoint queue is 234 Kb large. Assuming that we have a 512-ns round-trip delay which is 100 times long as the time of a 512-bit long packets transmitting through the data link. The result in this condition is shown in Figure 7.

Figure 7(a) shows that PFRR has much lower packet loss rate than the other four's. It only begins to drop about 10^{-3} packets while the input load reaches 99.5%, in the duration of our simulation. As we can see in Figure 7(b), while the input load grows higher than 60%, the delay of PFRR is much lower than the other four's. PFRR has the relative high delay to RRVIQ while under light load. This is because that there are many 'fake' cells to be switched. However, the delay is still low to 10^{-6} s level. The performance of all the other three flow-based methods



(a) Packet Loss Rate



(b) Average Delay

FIGURE 7 PFRR compared with other methods implemented with physical-wire back pressure in PPS architecture

with physical-wire back pressure in PPS architecture is worse than that of PFRR, both under heavy load and light load. Flow-based methods make all the cells of a same packet passing the same switching path to preserve the sequence. Assuming that each path works at the rate as $1/k$ of the input link, so a packet must take k times to be transferred through a certain path to the output, even if all other paths are free. That is the cause to bad delay performance in light load. However, cells in the same packet are dispatched to all paths uniformly both in PFRR and RRVIQ method, which offers a good load-balance and makes the delay much lower in light load.

As the load grows heavier, the back pressure controls of the other four methods start to work. Because the traffic distribution is highly skewed as shown in Figure 6, the back pressure flow control will greatly degrade the performance as shown in Figure 7. Although we reserved a separate buffer for each flow in the demultiplexer of the four methods, many flows still contend one output queue in the central OQ SM. That will lead to the saturation tree problem mentioned before. As shown in Figure 7, the other four methods' performance deteriorate with the input load increasing. On the contrary, there is no any closed-loop flow control scheme in PFRR, so the loss rate and delay grow moderately with the input load increasing. In addition, the waiting and padding process depicted in Algorithm 1 greatly smoothens the burstiness and skewness of the input load, which has been already proved in [4] and [22].

5.2.2 | Comparison with schemes with data plane back pressure

Next, we compare the packet loss rate and average delay of our method PFRR in CQPPS, with the other three flow-based methods in OQPPS architecture implemented with *data-plane* back pressure control cells. We assume that back pressure signals are transmitted by special control cells produced by SMs. With this implementation, if there is a congestion notification or flow control cancellation in an SM at one time slot, it has to send a control cell to each input demultiplexer and will not transmit data cells. All the other experiment configurations are the same as the previous simulations. The result in this condition is shown in Figure 8.

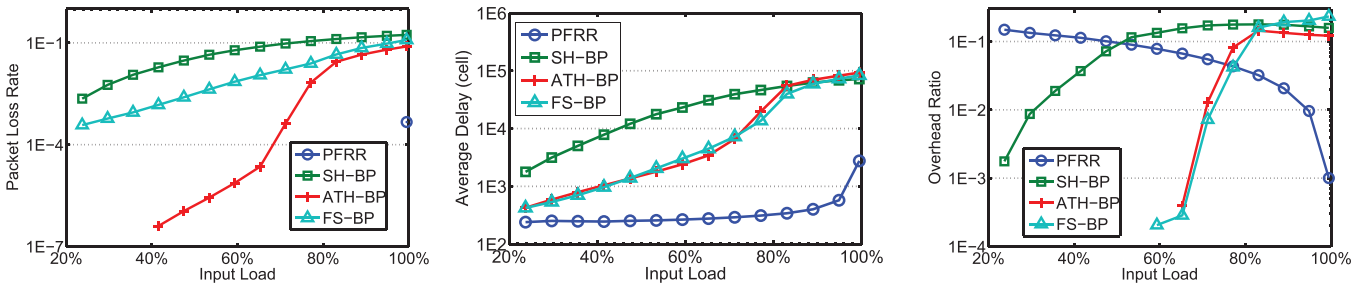
As shown in Figures 8(a) and (b), the three methods have the similar performance as before. However, with heavy loads, they perform worse than the physical-wired back pressure. That is because the control cells occupy data channels frequently while the input load is very high. We can see much clearer in Figure 8(c). This figure shows the padded cells transfer rate of PFRR compared with the control cells transfer rate of other methods. The control cells transfer rate is defined as the ratio of *the number of control cells that has been sent* to *the number of data cells that has been switched*. And the padded cells transfer rate in PFRR has the same definition as control cells transfer rate. And the cells which are switched to the output but cannot be used to reassembled to an integral packet are also counted into the number of padded cells. As the figure shows, PFRR needs less padded cells as the input load grows because it takes less time to collect the required amount of cells in the same flow. However, at the load around 60%, ATH and FS methods begin to transmit large amount of control cells due to congestion. That makes their performance degrade dramatically as shown in Figures 8(a) and (b).

5.2.3 | Comparison with flow-based methods in Clos Architecture

Furthermore, we compare our scheme to the other three flow-based methods in Clos architecture implemented with physical-wired back pressure as well. We set the Clos architecture with four input modules (IM), eight central modules (CM) and four output modules (OM), and make each modules with OQ buffering scheme. Considering the hardware resource we used in the former PPS SMs, we make each CM module in Clos work with 50 Gbps link rate. That makes each CM's switch capacity of 200 Gbps, the same as the SM's in former simulation. The rest configurations are the same as the first simulation. Figure 9 shows that the flow-based methods have a much better delay than in PPS architecture, because each path bandwidth is four times wide as before. However, the highly contention of output queue in Clos IMs, CMs and OMs makes the performance degrade dramatically when the input load grows higher as shown in the figure. Contrarily, PFRR avoids this problem without using the closed-loop flow control.

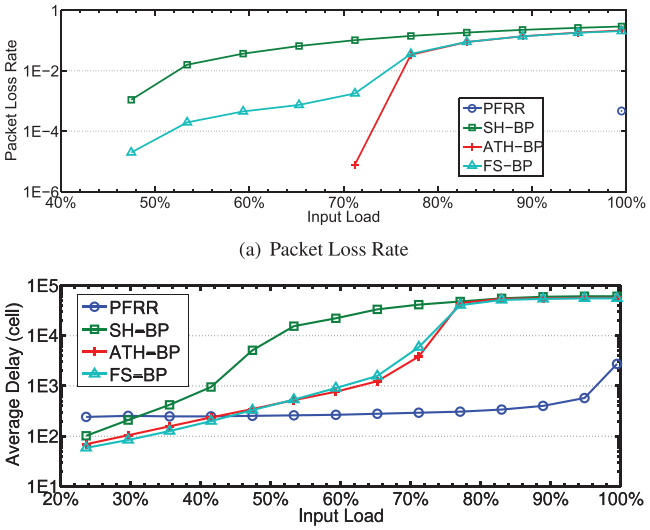
6 | DISCUSSION

In our scheme, K (the number of SMs) cells in the same round will either all pass or be dropped by all SMs. So the performance of our scheme is related to K and the packet length distribution.



(a) Packet Loss Rate (b) Average Delay (c) Overhead Ratio

FIGURE 8 PFRR compared with other methods implemented with data-plane back pressure in PPS architecture



(a) Packet Loss Rate

(b) Average Delay

FIGURE 9 PFRR compared with other methods implemented with physical-wire back pressure in Clos architecture

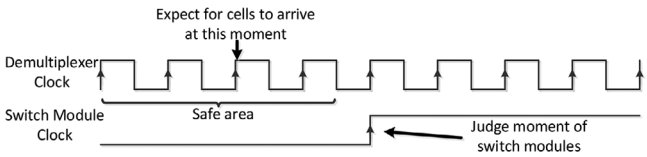


FIGURE 10 Schemes to ensure the simultaneousness of cells arriving at all the switch modules

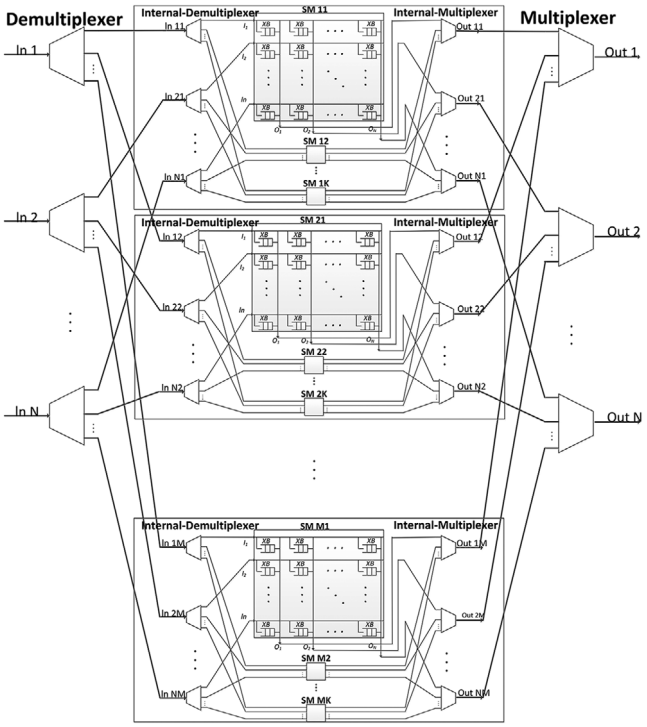


FIGURE 11 The scalability of CQPPS switch fabric

When the switch fabric scales to a very large port number and high link rate, there may be a large number of SMs and K is very large. For such a situation, we could simply divide the SMs into several small groups and reduce the waiting time to collect a round of cells to dispatch in the multiplexers. Suppose we wanted to implement a CQPPS structure with $N = 1024$ input ports and the link rate of each input port is 100 Gbps. Assuming that our chip only supports eight 100 Gbps input ports and eight 100 Gbps output ports. Then we can implement the architecture of 1024×100 Gbps input using the structure shown in Figure 11. As this figure shows, There is $N = 1024$ Demultiplexers and Multiplexers and each of them has a 100 Gbps input and output. Then there are $M = 16$ groups and each group has 1024 input ports (Internal-Demultiplexers) connected to the 1024 Demultiplexers and 1024 output ports (Internal-Multiplexers) connected to the 1024 Multiplexers. Therefore, the link rate between a pair of Demultiplexer and Internal-Demultiplexer is $100/16 = 6.25$ Gbps. In each group, there are eight SMs and each of them has a 1024 input ports connected to the 1024 Internal-Demultiplexers and 1024 output ports connected to the 1024 Internal-Multiplexers. So, each Internal-Demultiplexers connected to eight SMs and the link rate between a pair of Internal-Demultiplexer and SM is 6.25 Gbps/8 = 781.25 Mbps. Therefore, the total output link rate of one SM is 781.25 Mbps $\times$1024 = 800 Gbps (equal to a chip output capacity). So, this is a 1024×1024 switch fabric with a 1024×100 Gbps input link rate constructed by 128 chips which

has 8 × 100 Gbps input link rate. Also, we could make different divisions according to the packet length distribution under different traffic. How to set the number of cells in a round to get the optimal performance under different traffic is beyond the scope of this paper. However, this is worthy of study and could be one of our future work.

By uniformly spreading the cells of a same flow to all SMs simultaneously and use the same clock to drive all SMs, they are expected to behave exactly the same at most time, because the links are physically same. All SMs work according to the same clock and the moment of making scheduling decisions are exactly the same. However, the moment cells arrived at the inputs of different SMs may be slightly different (perhaps one or two cycles of the demultiplexer clock) due to minor transmission time difference and cause different behaviour of SMs. We could avoid this situation by utilizing the fact that SM's clock is slower than that of demultiplexer, and by limiting the expected arrival moment as Figure 10 shows. Assume all SMs do the scheduling at each rising cycle of SMs' clocks. Because the clock frequency of central SMs could be $1/K$ of input demultiplexers, we can make all expected cells arrival time at the middle of the low level of SM clock. First, at the very beginning when the system starts, all demultiplexers and SMs need a synchronization between each other. Then all demultiplexers know the clock frequency and phase of the SMs. Thus, by considering the transmission delay in the link, multiplexers could expect the right time to dispatch cells for every K cycles, to ensure that all cells arrive at the SM at the middle of the low level of SMs' clock. Thus cells arriving at the 'safe' area should be the same to all SMs, even though their exact arrival time can be slightly different.

Furthermore, a notification mechanism between SMs could be added to deal with the situation if any intermediate input link fails. Then certain actions can be taken by other SMs to ensure the follow-up scheduling behave the same. This mechanism only needs communication between the SMs thus has low delay. In reality, link error will rarely occur as technology develops.

We believe it is feasible to implement the aforementioned mechanisms to deal with slight difference in cell arrival time at different SMs, and to deal with link errors. Due to space limitation, we leave a more thorough study on these mechanisms as our future work.

7 | CONCLUSION

This paper proposed CQPPS, a highly scalable switch fabric, supporting simple schedule schemes without the need of closed control schemes. Based on the CQPPS architecture, we present a simple schedule scheme named PFRR. To the best of our knowledge, we propose a new method which can reach good performance with no need of closed-loop flow control. Our solution totally avoid the limitations of closed-loop flow control schemes in multi-path switch fabrics and greatly reduce the complexity of engineering implementation. By allowing potential cell loss in the switch fabrics and slightly higher light-load delay, CQPPS achieves loss rate orders of magnitudes lower than back pressure schemes, and high-load delay 10 times less than back pressure schemes. It also greatly reduce the complexity of engineering implementation.

ACKNOWLEDGEMENTS

This research was partially funded by the National Natural Science Foundation of China (No. 61872132), the Fundamental Research Funds for the Central Universities, Training Program for Excellent Young Innovators of Changsha.

ORCID

Qingqing Zhou https://orcid.org/0000-0003-4124-1495

REFERENCES

1. Cisco CRS carrier routing system 16-slot line card chassis enhanced router system description. (2018). https://www.cisco.com/c/en/us/td/docs/routers/crs/ec/system/description/lccecsysdescbook/lccecsysdesccards.html
2. Zbigniew, D., Grzegorz, R., Piotr, C.: MPLS-based reduction of flow table entries in SDN switches supporting multipath transmission. Comput. Commun. 151, 365–385 (2020)
3. Sule, O.T., et al.: A split-central-buffered load-balancing clos-network switch with in-order forwarding. IEEE/ACM Trans. Networking 27(2), 467–476 (2019)
4. Elahi, M., Shahhosseini, H.S.: LTD-router: Low latency traffic distributed FPGA based router architecture using dedicated paths. In: Iranian Conference on Electrical Engineering (ICEE), pp. 1601–1606, (2018)
5. Xiangli, L., et al.: A dual-threshold credit-based flow control mechanism for 3D network-on-chip. In: International Conference on IC Design and Technology (ICICDT), pp. 1–4 (2019)
6. Pant, N.: Priority aware congestion control scheme using dual thresholds. In: First International Conference on Secure Cyber Computing and Communication (ICSCCC), pp. 610–613 (2018)
7. Lemeshko, O., et al.: Mathematical optimization model of congestion management, resource allocation and congestion avoidance on network routers. In: International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo), pp. 1–5 (2019)
8. Kadhum, M.M., Mat Zin, N.N.: Congestion control and traffic management in wireless sensor networks. In: IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON), pp. 0830–0836 (2019)
9. Intel Stratix 10 TX device overview. (2019). https://www.intel.com/content/www/us/en/programmable/documentation/jzw1474049428757.html
10. Eira, A., Pedro, J., Pires, J.: Traffic grooming policies under switching constraints in next-generation transport networks. IEEE/OSA J. Opt. Commun. Networking 9(1), A125–A134 (2017)
11. Pfister, G.F., Norton, V.A.: 'hot spot' Contention and combining in multistage interconnection networks. IEEE Trans. Comput. C-34(10), 943–948 (2012)
12. Kuszmaul, B.C., Kuszmaul, W.H.: Avoiding tree saturation in the face of many hotspots with few buffers. In: IEEE Intl Conf on High Performance Computing & Communications, IEEE Intl Symp on Cyberspace Safety & Security, IEEE Intl Conf on Embedded Software & Syst (2014)
13. Gong, L., et al.: QPS-r: A cost-effective iterative switching algorithm for input-queued switches. In: VALUETOOLS '20: 13th EAI International Conference on Performance Evaluation Methodologies and Tools (2020)
14. Kanizo, Y., Hay, D., Keslassy, I.: The crosspoint-queued switch. In: IEEE INFOCOM 2009, pp. 729–737 (2009)
15. Keslassy, I.: The load-balanced router. PhD Thesis, Stanford University, (2010)

16. Gao, Y.: An adaptive scheduling algorithm for multi-priority traffic in load-balanced switch. In: International Conference on Information Science and System, pp. 200–213 (2018)
17. Zheng, Z., et al.: A concurrent round-robin-based variable-length packets dispatching scheme for satellite onboard clos-network switches. In: 8th International Congress on Image and Signal Processing (CISP), pp. 1510–1514 (2015)
18. Cao, Z., Wang, Z., Zegura, E.: Performance of hashing-based schemes for internet load balancing. In IEEE INFOCOM 2000, vol. 1, pp. 332–341 (2000)
19. Shi, L., et al.: Load-balancing multipath switching system with flow slice. IEEE Trans. Comput. 61(3), 350–365 (2012)
20. Aslam, A., Christensen, K.: Parallel packet switching using multiplexors with virtual input queues. In: IEEE LCN 2002, pp. 270–277 (2002)
21. Iyer, S., Awadallah, A., McKeown, N.: Analysis of a packet switch with memories running slower than the line-rate. In IEEE INFOCOM 2000, vol. 2, pp. 529–537 (2000)
22. Jaramillo, J.J., Milan, F., Srikant, R.: Padded frames: A novel algorithm for stable scheduling in load-balanced switches. IEEE/ACM Trans. Networking 16(5), 1212–1225 (2008)

How to cite this article: Pan B, Zhou Q, Chen G. CQPPS: A scalable multi-path switch fabric without back pressure. IET Commun. 2021;1–10.
https://doi.org/10.1049/cmu2.12236