A Parallel Packet Switch Architecture with Input-output-queued Switches and Buffering in the Demultiplexors

YI DAI, ZHI-GANG SUN, JIN-SHU SU
Institute of Network and Information Security
National University of Defense Technology
Changsha, Hunan, 410073
CHINA

Abstract: - A packet switch with parallel switching planes is a parallel packet switch (PPS). A PPS can scale-up to larger aggregate capacity and faster line speeds than can a single plane. It is an open problem to design a PPS that is feasible to implement using multiple lower speed packet switches. Many solutions proposed previously are essentially impractical because of high communication complexity. In this paper, we present a high performance PPS architecture by using a small fixed-size buffer in the demultiplexer and by applying the same matching at each of the $k$ parallel CIOQ switches during each cell slot. Our scheme guarantees a way for cells of a flow to be read in order from the output queues of the switches, thus, eliminating the need for cell reordering in multiplexor. Each multiplexor only need to deliver cells from the output queues of the $k$ parallel switch planes in round robin manner, and little state information easily obtained by the scheduler is communicated to the multiplexors. Our work in this paper reduces the communication overhead considerably and makes the PPS more practical to implement compared to other PPS designs.

Key-Words: - CIOQ switches, Packet switching, Parallel packet switch.

1 Introduction

As line rates increase beyond OC192 (10 Gb/s) to OC768 (40 Gb/s) and even to OC3072 (160 Gb/s), the development of faster memory and logic technology cannot keep up with the development of high-speed networks. In the past few years, the speed of DRAM has increased by 10% every 18 months while line rate increased 100% every 7 months [1]. However the widely used switch architectures need buffering packet, for example, OQ switches require buffer memory that operates at $N$ times the line rate, where $N$ is the number of ports of the switch, even in input-queued (IQ) and combined input-and-output queued (CIOQ) switches, memory operates at the same speed as the external line rate. So conditional switch architectures can neither overcome the memory bandwidth limitation nor operate as fast as line rates.

Parallel packet switches (PPS) have been studied in the past six years as a means of reducing memory bandwidth and scaling-up switch speeds beyond that of single-plane switches[2,3,4,6,7,8]. The PPS consists of identical lower speed packet switches operating in parallel. Arriving traffic is demultiplexed (spread) over the $k$ identical switches, switched to the correct output, and multiplexed (combined) before departing from the PPS. The key problem in a PPS is how to maintain the order of cells with less communication overhead so that in-order cell delivery is practical to implement. Unfortunately most multiplexing algorithms are impractical because of communication complexity and considerable state information required to maintain. In this paper we propose an efficient PPS architecture that uses CIOQ switch planes (where each plane runs, or switches, at $1/k$ the line speed) under the control of a single scheduler. Our scheme proposed in this paper reduces the communication overhead considerably, and cell resequencing at multiplexor become practical. The idea of using CIOQ switches as central stage of PPS and applying the same match at each switch planes was first proposed in Ref. [2] but its internal switch planes run at line speed $R$ instead of $R/k$, and its main motivation is to eliminate the need for speed up required by most practical switching algorithms regardless of using PPS that run at a speed slower than the line speed. The resequencing techniques used by Ref. [2] need considerable communication, upon applying a matching, the demultiplexer communicates to the scheduler the number of cells that will be forwarded, and the index of the switch plane that contains the oldest cell, and hence the multiplexor knows how many cells to read from the correct switch plane output queue. In contrast, the PPS developed in this paper neither each demultiplexor nor the scheduler need maintain any state information pertaining to the distribution of arriving cells in virtual output queues (VOQ) of the $k$ switch planes for each output.

In this paper, we reduce the communication overhead required to guarantee in-order cell delivery
by using round robin dispatch algorithm in the demultiplexors, \(k\)-parallel switching algorithm in the scheduler, and round robin re-assembly operation in the multiplexors. Compared with the previous work in PPS, our PPS design has the following distinct advantages:

1. The round robin dispatch algorithm distributes traffic equally among the switch planes, allowing the buffers in the switch planes to be utilized equally.
2. By using CIOQ switches without speedup as internal switch planes, the speedup requirement for internal switch planes is removed.
3. Cell re-assembly operation at the multiplexors becomes very simple to implement, because of less communication overhead.

2 PPS Architecture

The architecture of our PPS consists of the \(N\) input ports each having a demultiplexor with a small fixed-size buffer running at the line rate, and the \(N\) output ports each having a bufferless multiplexor. The center stage consists of \(k\) CIOQ switches in parallel; Fig. 1 depicts the traditional architecture of a CIOQ switch where all queues are first-in-first-out (FIFO) queues. If these CIOQ switch planes are allowed to forward cells independently, however, it is difficult to control the order in which cells of the same flow emerge at the multiplexors. As shown in Ref. [2], this may lead to deadlock at the multiplexor (i.e., no output queue can be read without violating the order of cells of the same flow). In order to avoid the type of deadlock depicted in [2], we consider the following property.

**Definition 1** (In-Order queuing): For every multiplexor \(M_j\), whenever there are cells in the output queues for output \(j\), at least a HOL cell can be read by the multiplexor \(M_j\) without violating the order of cells of a flow.

To proceed further, we define the following notation as in [2]:

- \((i, j)\) flow (of cells) from input \(i\) to output \(j\)
- \(C(i, j)\) a cell from input \(i\) to output \(j\)
- \(VOQ_j\) VOQ in switch \(l\)
- \(OQ_j\) output queue \(j\) in switch \(l\)

Fig. 2. Architecture of a PPS based on CIOQ switches

As shown in Fig. 2, each port is connected to all the CIOQ switch planes. Since the cells from each external input, of line rate \(R\), are spread over \(k\) links, each internal input link must run at a speed of at least \(R/k\). Since each multiplexor aggregates cells from \(k\) output queues in each switch plane to an external line rate of \(R\), both the output queues and the internal output links must operate at a speed of at least \(R/k\) to keep the external line busy. In general, we will use internal links that operate at a rate \(S(R/k)\), where \(S\) is the speedup of the internal link. Then we have the following definitions:

**Definition 2** (External Cell Slot): Refers to the time taken to transmit or receive a fixed length cell at link rate of \(R\).

**Definition 3** (Cell Slot): This is the time taken to transmit or receive a fixed length cell at link rate of \(S(R/k)\), where \(S\) is the speedup of the internal link.

Unless otherwise mentioned, in the proofs that follow, we neither require any synchronization between cell arrivals and the operation of the parallel switches, nor do we require any synchronization between the \(k\) switches themselves, except that they all perform a matching by the end of a cell slot. Our problem is to find a series of collaboration algorithms efficient and practical to implement in-order cell delivery. The architecture in Fig. 2 suggests the following natural decomposition of the problem:

1. Demultiplexor algorithm: at every input \(i\), deciding which switch plane to send a coming cell to.
2. Switching algorithm: In each of the \(k\) parallel switch planes, at each cell slot, deciding on matching, i.e., which cells to forward across the switch.
3. Multiplexor algorithm: at each output \(j\), deciding which output queue of the \(k\) parallel switch planes to read a cell from.
3 Approach

To specify our approach, we will describe how we carry out the three steps outlined in section 2 (demultiplexer algorithm, switching algorithm and multiplexor algorithm). Furthermore, we proof that our PPS has sufficient buffering to guarantee loss-free operation. We also proof that our PPS guarantees in-order delivery of cells.

3.1 Demultiplexer Operation

The demultiplexer architecture is the same as in the distributed PPS of [3]. As shown in Fig. 3 each demultiplexer contains \( \mathcal{k} \) FIFOs of depth \( N \) cells, one for each switch plane. A demultiplexer decides which switch plane to send a cell to only based on the destined output port of each arriving cell which enables the demultiplexors to operate independently, eliminating the communication complexity. To distribute the incoming cells equally among the \( \mathcal{k} \) parallel switch planes, the demultiplexer follows a special demultiplexing strategy, which we call Round-robin demultiplexing, described below:

Round-robin demultiplexing: Demultiplexer \( \mathcal{i} \) keeps \( N \) separate pointers \( P_1, \ldots, P_{\mathcal{K}} \); one for each output. The value of each pointer is initialized to 1. If pointer \( P_j =1 \), it indicates that the next arriving cell destined to output \( j \) will be sent to switch plane \( i \). Before being sent, the cell is written temporarily into the FIFO \( Q(i, l) \) where it waits until its turn to be delivered to switch plane \( l \). When the link from demultiplexer \( i \) to switch plane \( l \) is free, the head-of-line cell (if any) of \( Q(i, l) \) is sent.

An example of decisions made by the independent demultiplexors is illustrated in Fig. 3. It shows a PPS with \( \mathcal{k}=3 \) switch planes and no speedup. The demultiplexor operate at a line rate of \( \mathcal{R} \) and send cells to the internal switch planes over links which operate at a line rate \( SR/K = \mathcal{R}/3 \). The six cells shown arrive at input port 1 during six consecutive external cell slots, half are destined to output 1 and half are destined to output 2. As shown in Fig. 3, the demultiplexor dispatches the six cells destined for each output to the \( \mathcal{k}=3 \) FIFOs in a round robin manner. Assume the first cell destined to output 1 arrives during external cell slot \( \mathcal{T}_1 \), then by the end of external cell slots \( \mathcal{T}_6 \), these cells forwarded to corresponding switch planes are shown shaded in Fig. 3.

As shown in Ref. [3], without the small memories in the demultiplexors which can buffer the cells temporarily before sending them to the switch planes, the input link constraint cannot be met, and hence the Round-robin demultiplexing strategy is infeasible. The input link constraint means a demultiplexer is constrained to send a cell to a specific switch plane at most once every \( [k/s] \) external cell slots[3].

![Fig. 3. Demultiplexor (K=3, N=2), the cells shown shaded have been forwarded to corresponding switch planes and others still wait in the FIFOs.](image)

**Theorem 1:** A PPS without speedup can put the Round-robin demultiplexing into practice and guarantee loss-free operation by using the buffering of size \( KN \) cells in the demultiplexors.

**Proof:** Theorem 1 can be proved by following the steps in Theorem 6 in Ref. [3].

3.2 Switching Operation

In this section, we adopt the same switching algorithm called \( \mathcal{k}-parallel \) switching as [2]. Differing from [2], we turn our attention first to consider \( \mathcal{k}-parallel \) switching in a PPS where all parallel switch planes run at \( \mathcal{R}/\mathcal{k} \) instead of \( \mathcal{R} \). Our focus here is to proof that Round-robin demultiplexing together with \( \mathcal{k}-parallel \) switching can satisfy in-order queuing property. So it is possible for each bufferless multiplexor to read a cell (if one is available) from the output queues without violating the order of cells within a flow. In Section 3.3 we explain in detail how the multiplexor determines the correct queue to read from during each cell slot.

**Definition 4 (k-Parallel Switching):** \( \mathcal{k}-parallel \) switching is that where during each cell slot, the switching algorithm computes only one matching \( \mathcal{M} \) and applies it in all \( \mathcal{k} \) parallel switch planes.

When \( \mathcal{k}-parallel \) switching is used, all the switch planes will be controlled by a single scheduler instead of applying a match independently. In order to obtain \( \mathcal{k}-parallel \) switching algorithms, we simply apply the matching that the existing switching algorithms for a single switch compute in all \( \mathcal{k} \) parallel switch planes.

**Lemma 1:** If Round-robin demultiplexing and
**k-parallel** switching are used, then for any flow \((i, j)\), by the end of a cell slot T, either \(VOQ_s\) in all switches have the same length, or starting from a switch, we can find a round-robin order on the switches, \(S_i\) to \(S_k\), such that there exists \(0 < l < K\), such that \(VOQ'_{i,l}\) is the last \(VOQ\) that received a cell \(C(i, j)\) by the end of the cell slot T, the length of any \(VOQ'_{i,l}\) for \(l < s \leq k\) is L, and the length of any \(VOQ'_{i,l}\) for \(0 < s \leq l\) is L+1.

**Proof:** The proof is by induction on the number of cell slots.

Base Case: The lemma is trivially true at a fictitious cell slot before the beginning of the first cell slot.

Inductive Step: Assuming that the lemma is true at the end of cell slot T, we will prove that it holds at the end of cell slot T+1. At the end of cell slot T, there exists \(0 < l < K\), such that \(VOQ'_{i,l}\) is the last \(VOQ\) that received a cell \(C(i, j)\). Then we know by the lemma that at the end of the cell slot T, if \(l = k\), \(VOQ_s\) in all switches have the same length; if \(0 < l < K\), starting from \(S_j\), we can find a round-robin order on the switches. **K-parallel** switching during time slot T+1 will decrease the length of all \(VOQ_s\) by the same amount (by either 0 or 1). Without loss of generality, let \(d\) cells arrive from external input port during cell slot T+1, where \(0 \leq d \leq k\), because the external input link operates at most \(k\) times faster than internal input link. Then by the **Round-robin** demultiplexing, these \(d\) cells must be sent in a round-robin fashion (as a consequence of the **Round-robin** demultiplexing) from \(VOQ'_{i,l}\) to \(VOQ'_{i,s}\) during cell slot T+1, where \(a = (l + d) \mod k\). So starting from \(S_{(l+1) \mod k}\), we can find a round-robin order on the switches, \(S_{(l+1) \mod k}\) to \(S_{(l+k) \mod k}\), such that there exists \(a\), such that \(VOQ'_{i,a}\) is the last \(VOQ\) that received a cell \(C(i, j)\) by the end of the cell slot T+1, the length of any \(VOQ'_{i,a}\) for \(l + d < s \leq l + k\) is L, and the length of any \(VOQ'_{i,a}\) for \(l < s \leq l + d\) is L+1. Particularly when \(l = k\) and \(d = k\), \(VOQ_s\) in all switches have the same length. Therefore, at the end of cell slot T+1, the lemma is still true.

**Corollary 1:** If **Round-robin** demultiplexing and **k-parallel** switching are used, then at the end of a cell slot, the lengths of \(VOQ'_{i}\) and \(VOQ'_{k}\) differ by at most 1 for any two switches \(l\) and \(s\).

Using Lemma 1, we can now prove the following lemma:

**Lemma 2:** If **Round-robin** demultiplexing and **k-parallel** switching are used, then for any flow, at the end of a cell slot, either all cells at the input side are in distinct switches or the k oldest cells at the input side are in distinct switches.

**Proof:** If at the end of a cell slot T, there is some \(VOQ\) that is empty, then by Corollary 1, \(VOQ\) has length at most 1 for all \(l\), and, hence, all cells at the input side are in distinct switches.

If at the end of a cell slot T, no \(VOQ\) is empty, then for the \(k\) oldest cells at the input side not to be in distinct switches, it must be that some \(VOQ\), say \(VOQ'_{i}\), contains two of the \(k\) oldest cells \(C_i\) and \(C_j\), and another \(VOQ\), say \(VOQ'_{j}\), contains a cell \(C_k\) that is not among the \(k\) oldest cells. Without loss of generality, \(C_i\) is the head of \(VOQ'_{i}\), and the length of \(VOQ'_{i}\) is 1, then by corollary 1 the length of \(VOQ'_{j}\) must be 2, both \(C_i\) and \(C_j\) are in \(VOQ'_{i}\), let \(C_k\) is the head of \(VOQ'_{j}\). Then we know by Lemma 1 that we can find a round-robin order on the switches by the end of the cell slot T, hence \(C_k\) must be old than \(C_j\), which is a contradiction.

Using Lemma 2, we prove the main result of this section.

**Theorem 2:** If **Round-robin** demultiplexing and **k-parallel** switching are used, then for every output \(j\), at the end of a cell slot, either \(OQ'_{j}\) is empty for all \(l\) or there exists a flow such that its oldest \(d\) cells are at the head of \(OQ'_{j}\) for some consecutive \(l\) in a round-robin fashion, where \(0 < d \leq k\).

**Proof:** The proof is by induction on the number of cell slots.

Base Case: The lemma is trivially true at a fictitious cell slot before the beginning of the first cell slot.

Inductive Step: Assuming that the lemma is true at the end of cell slot T, we will prove that it holds at the end of cell slot T+1. At the end of cell slot T, we know by the lemma either \(OQ'_{i}\) is empty for all \(l\) or there exists a flow \(flow(a, j)\) such that its oldest \(d\) cells is at the head of \(OQ'_{j}\) for some consecutive \(l\) in a round-robin fashion. Then during cell slot T+1, the oldest \(d\) cells of \(flow(a, j)\) should be read out from the head of \(OQ'_{j}\) for some consecutive \(l\). We know by
lemma2 that for any flow(i, j), at the end of a cell slot, either all cells at the input side are in distinct switches or the k oldest cells at the input side are in distinct switches, hence upon k-parallel switching its oldest d cells should be switched to distinct OQ_s (i,j) at the same time. Assume the oldest d cells of flow(i, j) were switched to OQ_j for some consecutive l, and the oldest d' cells of another flow(b, j) were next switched to OQ_k for some consecutive l, where 0 < a, b ≤ N. Hence during cell slot T+1 the oldest d' cells of flow(b, j) will be at the head of OQ_k for some consecutive l in a round-robin fashion, and the lemma will hold.

Corollary 2: If Round-robin demultiplexing and k-parallel switching are used, then the in-order queuing property is satisfied.

Proof: We know by lemma2 that for every multiplexor M_j, as long as all OQ_s are nonempty, it can read the oldest d cells of certain flow(i, j) from some consecutive OQ_s without violating the order of cells of a flow. Therefore, the in-order queuing property is trivially satisfied.

3.3 Multiplexor Operation

We have already proven that when using Round-robin demultiplexing and k-parallel switching, the in-order queuing property is satisfied. Therefore, it is possible for every multiplexor to always deliver cells from the output queues of the k parallel switch planes without violating the order of cells of a flow. In this section we mainly explain in detail how a multiplexor M_j determines the set of HOL cells of OQ_s which can be read during the next cell slot in round robin manner. This can be done in different ways. One way is to use a traditional resequencing technique based on arrival timestamp. Each demultiplexor must add a tag to each cell indicating the arrival time of the cell to the demultiplexor. During each cell slot, the multiplexor sorts all the HOL cells by the same flow, and read the earliest d cells of certain flow in round robin manner, where 0 < d ≤ k. This approach requires additional access to the output queues which we assume not possible given that no speed up is available. A more efficient approach described below:

1. Whenever the demultiplexor D_j is notified by the scheduler that the number of cells in VOQ_s of flow(i, j) becomes 0, the pointer P_j at demultiplexor D_j that corresponds to output j is reset to 1. This strategy assures that the oldest cell of a flow is always in the first switch plane.

2. For each output j, the scheduler maintains a FIFO list L_j to record the number of cells switched to output j upon applying a matching M (this is easy to determine since it is either all the cells or k cells by lemma2). Therefore, upon applying the current matching M, the scheduler adds a number p to the tail of the FIFO list L_j, and p is the number of nonempty VOQ_y's. Furthermore if p ≠ k, the corresponding demultiplexor D_j is notified by the scheduler of the index j of the output for which (i, j) ∈ M, hence the pointer P_j is reset to 1.

3. Since the oldest cell of a flow is in the first switch plane, then if p = k the multiplexor M_j simply reads the HOL cells from OQ_l^j to OQ_p^j. If p ≠ k, the scheduler communicates to the multiplexor M_j with the number p, and p at the head of the FIFO list L_j, hence, the multiplexor M_j knows how many cells to read in round robin manner during each cell slot and read the HOL cells from OQ_l^j to OQ_p^j. It’s obviously that if and only if an unmatched input without any VOQ whose length is more than 1 in the first switch plane then p ≠ k. In practice the probability of the occurrence of this situation is very little, hence the communication both between the scheduler and the demultiplexors and between the scheduler and the multiplexors is considerably reduced.

It is interesting to compare this technique with the re-assembly operation with Round-Robin Reset demultiplexing proposed in [2]. In our scheme, the scheduler communicates to the multiplexors only under the circumstance that there is an unmatched input without any VOQ whose length is more than 1 in the first switch plane. But in Ref. [2] it is needed to tag the cells that are forwarded across the first switch with p obtained by the demultiplexers during each cell slot, where p is the number of cells that are going to be forwarded by the demultiplexor. For each demultiplexor it is difficult even impossible to determine the number p upon forwarding the cells to the first switch plane.
4 Implementation Issues

Given that our main goal is to make the PPS especially practical, we now reexamine its complexity in light of $k$-parallel switching algorithm described above.

1. Demultiplexor

Each demultiplexor maintains a buffer of size $Nk$ cells running at the line rate $R$, arranged as $k$ FIFOs. Next we want to determine whether the buffer of size $Nk$ can be placed on chip. For example, if $N = 1024$ ports, cells are 64 bytes long, $k = 10$, and the switch planes are CIOQ switches, then the buffer is about 5 Mb per multiplexor. This can be (just) placed on chip using today’s SRAM technology, and so can be made both fast and wide [6]. However, for much larger $N$, $k$ or $C$, this approach may not be practicable[6].

Each demultiplexor keeps $N$ separate pointers $P_1, ..., P_N$; one for each output. The value of each pointer is initialized to 1. And each pointer $P_j$ can be reset to 1 by the scheduler.

2. CIOQ switch planes

The $k$-parallel switching algorithm only enforced in the first switch plane. In order to determine the number of cells which are going to be switched during current matching, every $VOQ_j^p$ attaches a tag to indicate the corresponding $vq$ is empty or nonempty.

The scheduler maintains a FIFO list $L_j$ for each output $j$ to record the number of cells switched to output $j$ upon applying a matching $M$, whenever $p \neq k$, the scheduler sends a communication request along with the parameter $p$ to the multiplexor $M_j$.

3. Multiplexor

Before reading the first cell from $OQ_j^p$, the multiplexor $M_j$ examines the communication request signal to determine whether the scheduler will launch a communication, if the communication request signal is asserted, it will receive the parameter $p$ and read the HOL cells from $OQ_j^p$ to $OQ_p$, otherwise it simply reads the reads the HOL cells from $OQ_j^p$ to $OQ_j^p$.

5 Conclusion

A new architecture for a PPS using small high speed memories in the demultiplexors and CIOQ switches as central stage was investigated. Simple and distributed cell dispatch and re-assembly algorithms are applied in our PPS; these algorithms can be feasibly implemented. In theory, it is proved in our paper that Round-robin demultiplexing together with $k$-parallel switching can satisfy in-order queuing property, hence without any additional memory in the multiplexor the in-order cell delivery is achieved simply by reading the HOL cells from the output queues in round-robin manner. Therefore, cell ordering is easily guaranteed in our PPS because of less communication overhead. In summary, we think of this work as a step toward building high-capacity switches in which cell out-of-order problem resulting from parallel switching is not the obstacle of the implementation of the PPS.

(This work was supported by the National Keystone Basic Research Program (also called 973) under Grant No. 2003CB314802.)

References: