Smooth Switching Problem in Buffered Crossbar Switches

Simin He[1], Shutao Sun[2], Wei Zhao[3], Yanfeng Zheng[1, 2], Wen Gao[1, 2]

[1]Institute of Computing Technologies, Chinese Academy of Sciences, Beijing 100080, China
[2]Graduate School of Chinese Academy of Sciences, Beijing 100039, China
[3]Compunicate Technologies Inc., Beijing 100081, China

smhe@ict.ac.cn, stsun@jdl.ac.cn, wzhao@cti.com.cn, yfzheng@jdl.ac.cn, wgao@ict.ac.cn

ABSTRACT

Scalability considerations drive the switch fabric design to evolve from output queueing to input queueing and further to combined input and crosspoint queueing (CICQ). However, few CICQ switches are known with guaranteed quality of service, and credit-based flow control induces a scalability bottleneck. In this paper, we propose a novel CICQ switch called the smoothed buffered crossbar or sBUX, based on a new design objective of smoothness and on a new rate-based flow control scheme called the smoothed multiplexer or sMUX. It is proved that with a buffer of just four cells at each crosspoint, sBUX can utilize 100% of the switch capacity to provide deterministic guarantees of bandwidth and fairness, delay and jitter bounds for each flow. In particular, neither credit-based flow control nor speedup is used, and arbitrary fabric-internal latency is allowed between line cards and the switch core.

Categories and Subject Descriptors

C.2.6 [**Computer-Communication Networks**]: Internetworking – *Routers*; C.4 [**Performance of Systems**]: *Performance attributes*

General Terms

Performance, Design, Algorithms

Keywords

Switch, Scheduling, Smoothness, Buffered Crossbar, CICQ

1. INTRODUCTION

Switches with an increasing number of faster ports are in high demand to cope with the ever increasing network traffic, and this scalability consideration drives the switch fabric design to evolve from output queueing (OQ) to input queueing (IQ) and further to combined input and crosspoint queueing (CICQ).

A CICQ switch is implemented as a buffered crossbar with virtual output queues at the inputs and limited buffers at the crosspoints. It is scalable owing to its two features: distributed buffering and distributed scheduling. However, few CICQ switches are known with guaranteed quality of service. Besides, credit-based flow control is predominantly used till now to prevent overflow in crosspoint buffers; as a result, the size of each crosspoint buffer shall be at least the product of the line rate and the fabric-internal round-trip time so as to achieve work-conservation. For multi-rack multi-terabit switches, which have large fabric-internal latency and high line rates, crosspoint buffers shall be quite large; this is a bottleneck to scalability of CICQ switches.

In this paper we propose a novel CICQ switch called the smoothed buffered crossbar or sBUX, with three contributions: first, the design objective is shifted from work-conservation to smoothness so as to reduce buffer consumption and increase predictability; second, credit-based flow control is replaced with rate-based flow control through a smoothed multiplexer sMUX, which breaks the crosspoint buffer scalability bottleneck; third, comprehensive quality of service guarantees of sBUX are obtained, showing that sBUX is almost an OQ switch in terms of performance.

2. SMOOTHNESS

There are n flows of fixed-size cells sharing a link of bandwidth r; each flow f_i has a reserved bandwidth r_i, $r_i > 0$ and $\Sigma_i\, r_i \le r$. This specifies an instance $(r; r_1, r_2, \ldots, r_n)$, which can be reduced to its normal form $(1; w_1, w_2, \ldots, w_n)$, abbr. $(w_1, w_2, \ldots, w_n)$, in which $w_i = r_i/r > 0$ and $\Sigma w_i \le 1$. Time is slotted, with slot t denoting the real interval $[t, t + 1)$, and slot interval $[t_1, t_2)$ denoting the slot set $\{t_1, t_1 + 1, \ldots, t_2 - 1\}$. A schedule is a mapping from slots to cells.

The smooth multiplexing problem (SMP) is to generate a *smooth* schedule such that cells of each flow are *smoothly* or *evenly* distributed in the whole sequence. Intuitively, in an ideally smooth schedule for an SMP instance $(w_1, w_2, \ldots, w_n)$, any interval of l consecutive slots should cover $(l \cdot w_i)$, or in practice, either $\lfloor l \cdot w_i \rfloor$ or $\lceil l \cdot w_i \rceil$ number of cells of flow f_i; in a complementary view, successive $(s+1)$ number of cells of flow f_i should be spacing (s/w_i), or either $\lfloor s/w_i \rfloor$ or $\lceil s/w_i \rceil$ number of slots apart.

Covering smoothness. Let $Cover_i(t, l)$ denote the number of cells of flow f_i that are scheduled by a scheduler inside slot interval $[t, t + l)$. Given an arbitrary slot interval $[t_1, t_2)$, $t_1 < t_2$, we measure the covering smoothness of the actual distribution of cells of flow f_i within this interval by the following two worst-case covering deviations from the ideal:

$$cvr\text{-}dev_i = max\{\lfloor l \cdot w_i \rfloor - Cover_i(t, l) \mid [t, t + l) \subseteq [t_1, t_2)\};$$
$$CVR\text{-}dev_i = max\{Cover_i(t, l) - \lceil l \cdot w_i \rceil \mid [t, t + l) \subseteq [t_1, t_2)\}.$$

Spacing smoothness. Let $Pos_i(j)$ denote the slot of the j-th cell of flow f_i scheduled by a scheduler, and let $Space_i(j, s)$ denote an s-step space between the positions of the j-th and $(j+s)$-th cells of flow f_i: $Space_i(j, s) = Pos_i(j+s) - Pos_i(j)$. Then given an interval $[Pos_i(j_1), Pos_i(j_2) + 1)$, $j_1 \le j_2$, we measure the spacing smoothness of the actual distribution of cells of flow f_i within this interval by the following two worst-case spacing deviations from the ideal:

$$spc\text{-}dev_i = max\{\lfloor s / w_i \rfloor - Space_i(j, s) \mid j_1 \le j, j + s \le j_2\};$$
$$SPC\text{-}dev_i = max\{Space_i(j, s) - \lceil s / w_i \rceil \mid j_1 \le j, j + s \le j_2\}.$$

Connection. As a further proof of rationality, covering and spacing smoothnesses are consistent and corresponding:

This work was supported in part by grants NSFC-69983008, KGCXZ-103 and 863-2001AA112100 and by a basic research grant of ICT.

SIGMETRICS'05, June 6-10, 2005, Banff, Alberta, Canada.
ACM 1-59593-022-1/05/0006.

THEOREM 1. *Given an SMP instance, then for any flow f_i,*
(1) $SPC\text{-}dev_i \le \lceil cvr\text{-}dev_i / w_i \rceil$, (2) $spc\text{-}dev_i \le \lceil CVR\text{-}dev_i / w_i \rceil$,
(3) $cvr\text{-}dev_i \le \lceil SPC\text{-}dev_i \cdot w_i \rceil$, (4) $CVR\text{-}dev_i \le \lceil spc\text{-}dev_i \cdot w_i \rceil$,
where the left-hand side of ≤ is defined within the interval in which the right-hand side of ≤ is defined.

3. SMOOTHED MULTIPLEXER sMUX

Given an SMP instance ($w_1, w_2, \ldots, w_n$). Besides the proportion w_i, each flow f_i is additionally associated with an *initiation time* I_i to mark the earliest time slot a scheduler is ready to schedule flow f_i. This is to reflect the dynamic scheduling of newly admitted flows in practical applications. Starting from time I_i, the j-th (j = 1, 2, 3…) cell service provided for flow f_i is *eligible* at time $e_{i,j} = I_i + (j-1)/w_i$ and is expected to finish before *deadline* $d_{i,j} = I_i + j/w_i$.

Algorithm sMUX: *At each time slot t, among those flows that are eligible for scheduling at time t, i.e., their cells to be serviced have eligible times no later than t ($e_{i,j} \le t$, equivalently, $\lceil e_{i,j} \rceil \le t$), allocate slot t to the flow with the earliest upper-rounded deadline $\lceil d_{i,j} \rceil$; ties are broken arbitrarily. If no flow is eligible, slot t is left idle.*

Actually sMUX allocates the j-th cell of flow f_i in $[\lceil e_{i,j} \rceil, \lceil d_{i,j} \rceil)$, and provides the guaranteed smoothness bounds as follows.

THEOREM 2. *Given an SMP instance ($w_1, w_2, \ldots, w_n$) with initiation times ($I_1, I_2, \ldots, I_n$) and a sMUX schedule. Then for any flow f_i,*
(1) $cvr\text{-}dev_i \le 1$, $CVR\text{-}dev_i \le 1$,
(2) $spc\text{-}dev_i \le \lfloor 1 / w_i \rfloor$, $SPC\text{-}dev_i \le \lfloor 1 / w_i \rfloor$.

The sMUX schedule for any SMP instance ($w_1, w_2, \ldots, w_n$) is proved to have a covering deviation for at most one slot or cell, which could be outperformed only by an ideal schedule if existing. In this sense the sMUX schedule is almost ideal or optimal. When an ideal schedule does not exist for the given problem instance, which is the normal case, the sMUX schedule could be considered optimal. Besides, the spacing jitter can be construed as delay jitter introduced by sMUX into each ideally arriving flow. Assume cells of flow f_i arrive at sMUX at eligible times I_i, $I_i + \lceil 1/w_i \rceil$, $I_i + \lceil 2/w_i \rceil$, …, with all spacing and covering deviations equal to 0. If sMUX only introduces a constant delay to each cell, then spacing deviations of leaving cells should remain zero. Therefore, any non-zero spacing deviations are actually the delay jitters, resulting from variable delays introduced to cells by sMUX.

4. SMOOTHED CICQ SWITCH sBUX

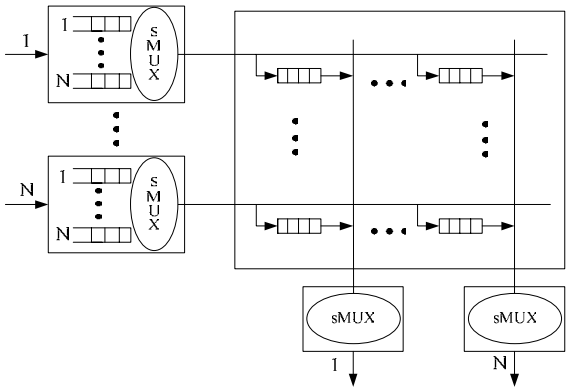

Figure 1. The architecture of CICQ switch sBUX.

The CICQ switch sBUX uses sMUX as input and output schedulers, which can be placed *physically* at anywhere in the switch fabric, either in the line cards or in the switch core. The fabric-internal latency can be arbitrary and variable for each connection between a line card and the switch core (see Figure 1). For each input-output pair (i, j), there is one flow $f_{i,j}$ of fixed-size cells, with a reserved bandwidth $r_{i,j}$, satisfying the admissible condition: $\Sigma_i r_{i,j} \le 1$ for all j and $\Sigma_j r_{i,j} \le 1$ for all i.

Crosspoint buffer occupancy. What's unique to switch sBUX is that between each virtual output queue and its corresponding crosspoint buffer, credit-based flow control is replaced by rate-based flow control through sMUX. Let's take the crosspoint buffers as our point of view, and in all practical cases, initiation of input scheduling never precedes that of output scheduling.

THEOREM 3. *The occupancy of any crosspoint buffer in sBUX never exceeds four cells, regardless of fabric-internal latency and line rate.*

Then a four-cell crosspoint buffer in sBUX guarantees no loss of cells.

Guaranteed performances. The sBUX switch core, including input and output schedulers and crosspoint buffers, when treated as a whole, has the following smoothness and delay guarantees:

THEOREM 4. *In a sBUX switch, flow $f_{i,j}$ is guaranteed a service with the following smoothness bounds:*
(1) $cvr\text{-}dev_{i,j} \le 3$, $CVR\text{-}dev_{i,j} \le 1$;
(2) $SPC\text{-}dev_{i,j} \le \lceil 3/r_{i,j} \rceil$, $spc\text{-}dev_{i,j} \le \lceil 1/r_{i,j} \rceil$.

THEOREM 5. *In a sBUX switch, once a cell of flow $f_{i,j}$ goes to the head of $VOQ_{i,j}$, it will wait at most $\lfloor 5/r_{i,j} \rfloor$ slots before it departs from the crosspoint buffer $XPB_{i,j}$ (excluding the constant fabric-internal propagation latency).*

In summary, a sBUX CICQ switch has almost the same quality of service guarantees as an OQ switch, yet with far superior scalability. To our knowledge, no other CICQ or CIOQ (combined input- and output-queued) switches have been proved to hold such *comprehensive* quality of service guarantees.

5. CONCLUSIONS

The key design objective of sBUX is smoothness rather than work-conserving-ness as most researches do. In principle, work-conserving-ness aims to fully exploit the link bandwidth, which was precious before but no longer now, yet at the cost of large buffer, which is the major bottleneck to scalability both now and in the foreseeable future. In contrast, smoothness aims at small buffer and increased predictability. In some sense, the difference between work-conserving-ness and smoothness can be traced back to the difference between packet switching in the Internet and circuit switching in telecom. Clearly sBUX adopts the latter approach, and realizes a harmonious unification of scalability and predictability.

The smooth switching problem, besides being novel, is actually a general and kernel problem for all switch fabrics. While it has been solved for the CICQ switch sBUX, the problem remains open both for the more traditional switch fabrics, such as IQ switches or CICQ switches with credit-based flow control, and for the newly proposed ones. We believe the concepts defined and the analyses conducted in this paper demonstrate that the study of the smooth switching problem is feasible and promising.