Closed Loop Credit-Based Flow Control with Internal Backpressure in Input and Output Queued Switches

Rainer Schoenen and Achim Dahlhoff
Institute for Integrated Signal Processing Systems
Aachen University of Technology, Templergraben 55, 52056-Aachen, Germany
schoenen@ert.rwth-aachen.de, http://www.iss.rwth-aachen.de/

Abstract

A credit-based flow control scheme is very effective for handling cells of the ABR or Controlled Transfer service class in ATM networks. The properties of immediate ramp-up after congestion and the impossibility of buffer overflow by construction are what makes this method very attractive. For best results, many switches along the route must support flow control, acting as virtual source/destination node. To support closed loops along the route each of these switches must also provide an internal flow control to pass the backpressure from the destinations back to the sources and avoid internal buffer overflow. In this paper we treat the interaction of external and internal flow control and the alternatives of internal backpressure mechanisms, emphasizing Virtual-Output-Queued switches. The properties are studied with Petri net models.

1 Introduction

Flow control for the ABR service in ATM networks has excited great debates in the past. Satisfactory mechanisms have rarely been proposed yet. Undoubtedly flow control must provide closed loops on the route from source to destination [1], where results become better the more intermediate switches (hops) actively contribute to the flow control (called virtual source/destination).

Switches supporting flow control have to regulate their cell stream output according to flow control messages from downstream switches. This leads to a controlled emission of cells, independent for each ABR connection. Towards the source they have to generate flow control messages themselves which aim at avoiding buffer overflow within this switch, inhibiting cell loss. But these mechanisms only work together if the sending component (downstream) and the receiving component (upstream) are coupled by any kind of internal flow control or backpressure mechanism (**fig. 1**). In an $M \times M$ switch (M =number of ports) this backpressure must be routed for any VC connection v from the exit port of v to the (different) input port of v. This must ensure to propagate the downstream backpressure correctly, i.e. without the chance of a fatal delay, and prevent all internal cell buffers from overflow.

The switch internal realization of flow control, i.e. the location of its components, the location of buffers and schedulers heavily depends on the underlying switch architecture. Future switches will more and more consist of the extremely powerful input queued architecture, because the access rate of crossbar and buffer memory is not higher than the line rate of the connected links. The head-of-line blocking problem is solved by using Virtual Output Queueing (VOQ) [2]. With VOQ each input port manages a separate logical queue for each output (**fig. 2**). This allows the maximum throughput of 100% [3, 4, 5]. The classical output-queued (OQ) architecture requires much higher relative access rates to the switch fabric and the internal memory. For uncontrolled traffic, the buffer location is in the input ports with VOQ, but in the output port in the OQ configuration. As a consequence, avoiding buffer overflow must be taken care of around this usual memory location. Otherwise secondary queues build up at new points where congestion is sensed, e.g. in the output port with VOQ.

Credit-based flow control (CBFC) as a special case has a deterministic behavior in each flow control loop where buffer overflow is impossible by construction [6, 7]. Building upon such a mechanism we pro-

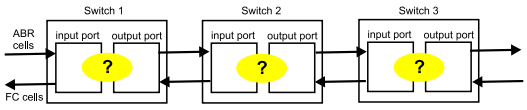

Figure 1: 3 switches with flow control loops

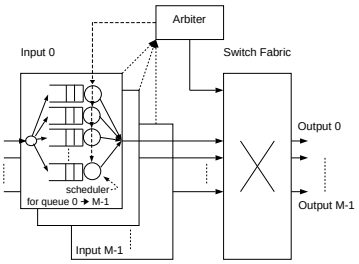

Figure 2: virtual output queueing

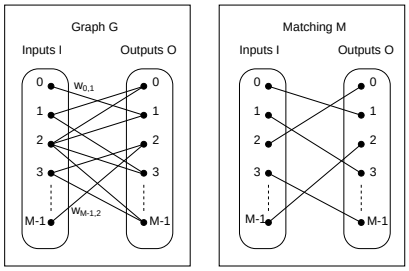

Figure 3: Bipartite graph matching

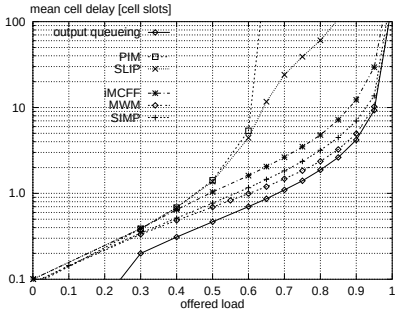

Figure 4: $E\{d\} = f(\rho)$ for VOQ

pose and compare several variants of internal backpressure with methods that have been used previously for the external flow control evaluation [7]. The Petri net (PN) paradigm [8] is ideally suited to model CBFC because of the affinity of cells and credits with tokens. In this paper we give models for each alternative and study their properties.

The paper is organized as follows. Section 2 discusses the VOQ switch architecture. In section 3 the external credit-based algorithm is explained, followed by the proposed backpressure schemes in section 4. The analysis based on Petri nets is presented in section 5.

2 VOQ and Arbitration

An OQ switch architecture has an internal interconnection of sufficient speed to let all cells queue in the output ports immediately. In contrast, the VOQ configuration [2] shown in fig. 2 consists of M ports for input and output, a nonblocking switch fabric and an arbitration unit. Arriving cells on input port i are placed into the corresponding logical[1] queue for their destination port o. In each time slot the arbiter selects unique pairs of input and output ports (a "match" (i,o)) based on information sent to it from the input ports. It has been shown that 100% throughput can be achieved for all admissible i.i.d. arrivals [4] with proper algorithms based on bipartite graph matching [2] (**fig. 3**). Practical algorithms exist with the unweighted PIM [9], iSLIP [3], WFA [10] or the weighted (better performing) iMCFF [5] or SIMP [11]. Its performance is shown in **fig. 4** for renewal traffic. A method to enforce a global priority scheme is shown in [12]. Relevant are also arbitration with internal speedup, e.g. [13], and Explicit-Rate Flow control for VOQ switches [14].

These architectural differences have a great influence on the way how internal flow control must be organized. As it will be shown, the position where most of the cells are queued in front of a bottleneck is usually the best position also for the flow control unit (**fig. 6**).

[1] the real queue organization here is per-VC

3 Credit-Based Flow Control

In this paper a CBFC realization is assumed similar to QFC [15] (CT [16]) or FCVC [6]. For the basic operation each link is considered to be a closed loop between adjacent switches (called **S** and **R** here). For each loop, the receiving switch **R** has an available total buffer'of size l_l ($limit_{link}$). Logically this buffer is associated with the input port i connected to the controlled link. This buffer is partitioned into individual memory l_v ($limit_v$) for each connection v out of C_i ABR connections through that input port i. This partitioning is only logical and can be overlapping, i.e. buffer sharing is possible with $\sum l_v > l_l$. Both buffers are protected by separate flow control loops. The sender **S** knows about each limit l_v (l_l) and installs this as the initial credit c_v (c_l). Cells may now be sent arbitrarily (after the scheduler has served all higher-priority cells) as long as the current credits c_v (c_l) are positive. For each sent cell a counter of transmitted cells t_v is incremented (from zero initially). **R** increments a counter r_v of cells that are received and forwarded to the next hop. In regular intervals (each $N2_{VC}$ values) this counter is sent back to the source **S** which updates its state for all included connections v by replacing the previous values of forwarded cells, f_v, with the new contents f_v^{new} of each record. The current credit for connection v (and similar for the link l) is then

$$credit_v = c_v = l_v - t_v + f_v \geq 0 \qquad (1)$$

$$credit_l = c_l = l_l - t_l + f_l \geq 0 \qquad (2)$$

In [7] we have introduced a Petri net model which includes all necessary details of one flow control loop. Here we extend the model to contain multiple connections, a link flow control (eq. 2) and switch internal communications.

As a short Petri net introduction (see [8] for more), *places* (circles) model storage locations for *tokens*. Its current contents, *marking*, defines the state of the system. *Transitions* (bars) model actions; when they *fire*, a specified number of tokens is removed from all its

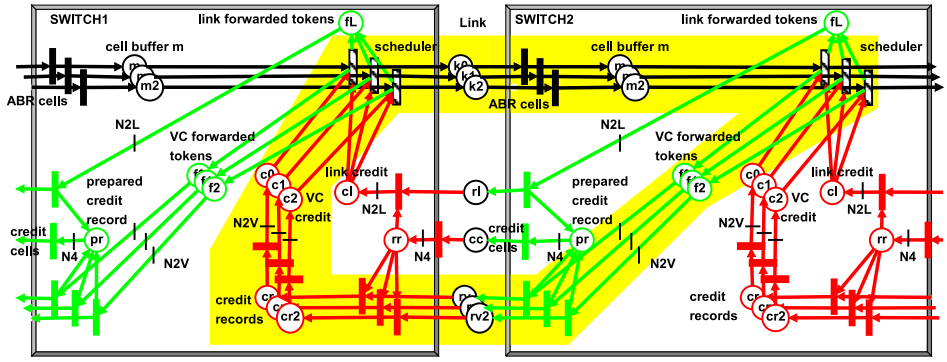

Figure 5: PN system for two credit-based switches **S** and **R** (one flow control loop)

input places and all output places get additional tokens. The number annotated on the arcs specifies this number. Inhibitor arcs (circle instead of arrow) prevent the transition from firing if there is at least one token in the adjacent place. Enabling arcs (arrows in both directions) let the transition fire if there is at least one token, but this token is not removed. In stochastic Petri nets [17], transitions have associated execution time distributions distinguished by their pattern. Small filled bars mean immediate execution.

The model for two switches and one flow control loop is shown in **fig. 5** and is subject to structural analysis in this section. It shows the components seen globally, without internal flow control. Up- and downstream fractions of the flow control loops are distinguished by their color. The main components in the cell flow in the forward direction are the cell buffer memory m_v and the cells in flight on the link in k_v. Cells may only be sent by the scheduler if there are tokens in the credit places c_v. In backward (feedback) direction *forwarded tokens* queue in f_v until a record in pr is complete. $N4$ of such records build a credit update cell in cc, which is again decomposed in the upstream switch into records in cr_v. rv_v exists for distinguishing the connections. There is one *VC* flow control loop per VC (here $C = 3$ connections), using the feedback quantization constants $N2_{VC}$ (N2V) discussed in [15, 7] to decrease the feedback frequency. Each link is also controlled by a *link* flow control loop (f_l, r_l, c_l), quantisized with $N2_{link}$ (N2L). The packing into a feedback cell consisting of $N4$ records reduces the flow control bandwidth overhead, which is discussed into details in [7].

Let this system be implemented in each hop and the connection between them be made by arcs which have buffering places to hold tokens which are currently on the fly. In fig. 5 the closed loop for three connections is emphasized (eq. 3). The consistency of each loop implies [18] that the weighted sums of tokens (eq. 3-6) on them are constant:

$$\forall v : k_v + m_v + f_v + N2_{VC} \cdot (r_v + cr_v) + c_v = l_v \quad (3)$$

$$\sum k_v + \sum m_v + f_l + N2_{link} \cdot r_l + c_l = l_l \quad (4)$$

$$\sum (k_v + m_v + f_v + c_v) + N2_{VC} \cdot (\sum cr_v + rr + pr - rl + N4 \cdot cc) = \sum_v l_v \quad (5)$$

$$\sum (k_v + m_v) + f_l + c_l + N2_{link} \cdot (pr + rr - \sum rv_v + N4 \cdot cc) = l_l \quad (6)$$

Therefore there cannot be more tokens in any place of the loop than there are initial credits ($c_v = l_v$, $c_l = l_l$) in the credit places. Thus the buffer places m_v can also at most contain l_v cells each and the whole buffer memory usage l_{max} is bounded by both the sum of all relevant connection limits and the link limit.

$$l_{max} := \max(\sum m_v) = \min(\sum_{\forall v} l_v, l_l) \quad (7)$$

The boundedness of buffer usage implies that no cell loss can occur due to buffer overflow. For the discussion of dimensioning and quantization effects see [7]. The option to reduce l_{max} below the sum of the individual credits (”buffer sharing”) allows a tradeoff between memory and peer blocking probability [7]. Peer blocking appears when the total buffer is filled with cells of some connections and its link credit is $c_l = 0$, but some other connections have individual credit $c_v > 0$ left. This phenomenon was also called ”hot-spot” problem in [19]; however it can only happen if $c_l < \sum_{\forall v} c_v$.

The timing behaviour is determined by the type of the transitions. Except the ”schedulers” all of them are immediate and do not contribute to the latency. The sources of latency are the link propagation delay and the waiting for multiple-weighted input arcs to gather enough tokens for the adjacent transition to fire [7].

4 Internal Backpressure – Principles and Problems

So far the switch internal structure has been neglected. When it comes to an implementation, decisions have to

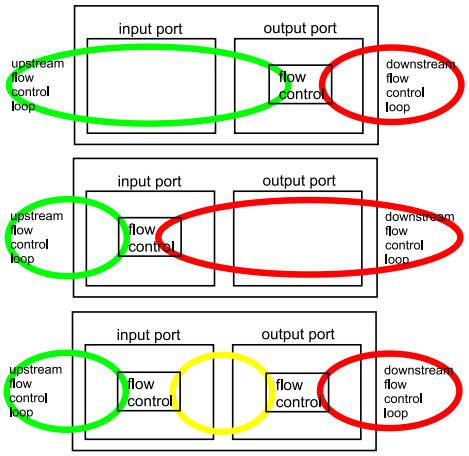

Figure 6: Internal flow control position (independent for connections or link)

TV	test VC FC inequality (eq. 1)
TL	test link FC inequality (eq. 2)
RFC	receive flow control cells
DRV	dispatch flow control records for VC
DRL	dispatch flow control records for link
HFV	hold forwarded counter for VC
HFL	hold forwarded counter for link
DFV	dispatch forwarded counter for VC
DFL	dispatch forwarded counter for link
MRV	make flow control records for VC
MRL	make flow control records for link
MFC	make flow control cells

Table 1: Internal components of CBFC

be taken where to position the pieces of the flow control (**fig. 6**). **Table 1** shows all the atomic components that have to be implemented. In this list, blocks for *dispatching* are responsible to communicate the messages to the recipient in the correct port(s). There are several ways to distribute these on input and output ports of a switch. The first important observation is that the methods for link and VC flow control are totally independent (except in the cell creation *MFC* and unpacking *RFC* blocks). Thus the partitioning decision (fig. 6) can be made independently for both. For each switch architecture an adapted internal flow control is proposed. The main flow control core connecting the loops in fig. 6 is the test of the flow control inequalities (1,2) and the corresponding on/off influence on the cell stream. At this point the group of transitions are called *scheduler*, because in each time slot only one of the eligible cells can be chosen for transmission by the scheduler. Its behavior must be static priority on the first level (for real-time and ABR traffic separation) and a rate fair algorithm within the ABR class. For Round-Robin (RR) this scheduler is shown in **fig. 7**. In any case the downstream and upstream loops must be closed, i.e. the forwarded counter (sent upstream as f_v^{new}; represented by places f_v) is incremented exactly in the instant when the cell is dismissed out of the upstream loop into the next loop. If the upstream-f_v is maintained in the input port, but the downstream-inequality is tested in the output port, an infinite number of cells may queue in front of the outputport scheduler. In this case an explicit internal flow control loop must be inserted to avoid buffer overflow (fig. 6 bottom).

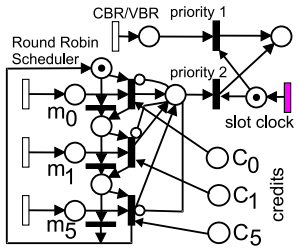

Figure 7: PN system for Round-Robin scheduler

The main degrees of freedom for the internal flow control realization manifest in the following:

- Two independent mechanisms (for VC or link),
- control in input port, output port or both.

Minor differences exist additionally with the methods for returning backpressure in backward direction (which determines the additional switch internal bandwidth required in that direction).
The consequences of each variant manifest in a number of ways:

Buffer requirements in input port depend on the maximum number of tokens in the input port memory places which are bounded by the loop equations of the upstream control loops.

Buffer requirements in output port depend on the maximum number of tokens in the output port memory places which are bounded by the loop equations of the control loops around these places. This can be the upstream, downstream or internal control loop (fig. 6).

Efficiency of buffer sharing is only acceptable if the total buffer per port is limited by l_{link}, and this amount of memory is only necessary in either input or output port. With the worst mechanism, l_{link} is needed in the input port, but $\sum l_v$ in the output ports.

The hotspot problem means that all connections using one specific input port are reduced in rate due to depleting uplink credits, which protect this filled input port from overflow [19]. The reason for this full input port may be only one congested

flow control position	$\max_{VC}(\sum_{v\in V_i^i} m_v^i) =$	$\max_{VC}(\sum_{v\in V_o^o} m_v^o) =$
input	$\sum_{v\in V_i^i} l_v^u$	$\sum_{v\in V_o^o} l_v^o$
output	$\sum_{v\in V_i^i} l_v^u$	$\sum_{v\in V_o^o} l_v^u$
i+o	$\sum_{v\in V_i^i} l_v^u$	$\sum_{v\in V_o^o} l_v^x$

Table 2: Variants of internal flow control (VC)

flow control position	$\max_{link}(\sum_{v\in V_i^i} m_v^i) =$	$\max_{link}(\sum_{v\in V_o^o} m_v^o) =$
input	$l^u_{link=l}$	$l^d_{link=l}$
output	$l^u_{link=l}$	$\sum_{\forall l} l^u_{link=l}$
i+o	$l^u_{link=l}$	$l^x_{link=l}$

Table 3: Variants of internal flow control for links

output port, for which the sum of all l_v from this input port is bigger than l_{link}. In this case innocent connections are affected and the link utilization is unnecessarily reduced. This problem is similar to head-of-line blocking in input-queued switches. However, this only appears if VOQ *and* intensive buffer sharing is used. In [7] an alternative buffer dimensioning has been proposed, which does not require buffer sharing.

Fairness is the concern that each capable connection gets an equal share of the spare bandwidth. This is usually addressed by a proper (RR) scheduler in the output port for ABR. However, with VOQ there is usually no freedom of choice in the output port because all cells queue in the input port. In this case arbiter and input schedulers are responsible for maintaining fairness.

Cooperation with arbitration means that the correct input port state information is available in the arbiter, e.g. the number of eligible cells is zero when flow control in the input port stops all connections, even if there are cells queued. With weighted arbitration algorithms [11], where w bits are used for the queue weight, another issue is that good fairness ($w = 1$) is contrary to best delay performance ($w = 3$).

Especially the buffer memory bounds differ significantly among the methods. Ideally the buffer must be limited by equation 7 and only input or output port must be used for buffering. The following section treats our proposed backpressure mechanisms and their properties related to these problems.

5 Internal flow control

Crucial for all variants of internal flow control is the position within the switch where the flow control test (eq. 1 for connections and 2 for the link) is implemented: In the inputport, outputport or both. The notation (VC:location,Link:location) is used to denote the location of these flow control tests. In **table 2** and **3** the resulting variants are summarized with their buffer memory properties that have been derived from their corresponding loop equations of their Petri net models given in section 5.1. The following variables are used: The contents of the queue of connection v is m_v^i in the input port and m_v^o in the output port. V_i^i is the set of connections flowing through input port i, similar with V_o^o for output port o. For the upstream link l_v^u and $l^u_{link=l}$ and the for the downstream link l_v^d and $l^d_{link=l}$ are given. Upstream and downstream limits are usually very different, because their magnitude depends on the connected link length and rate [7]. For the internal loops, if used, additional limits l_v^x and $l^x_{link=l}$ are introduced, where $link = l$ corresponds to output port l. Note that $l_v^x \ll l_v^d \approx l_v^u$ and $l^x_{link=l} \ll l^d_{link=l} \approx l^u_{link=l}$. To determine the upper bound on the buffer usage, the results for the VC and link flow control constraints must be combined:

$$\text{input port:} \quad M^i_{max} = \min\left(\max_{VC} \sum_{v\in V_i^i} m_v^i, \max_{link} \sum_{v\in V_i^i} m_v^i\right) \tag{8}$$

$$\text{output port:} \quad M^o_{max} = \min\left(\max_{VC} \sum_{v\in V_o^o} m_v^o, \max_{link} \sum_{v\in V_o^o} m_v^o\right) \tag{9}$$

The final goal is that only the input or output port requires a buffer memory of the size l_{max} given by eq. 7. Only few combinations really limit the total buffer size in the desired way. If no assumptions are made on the switch architecture and there is no explicit internal control loop, the flow control loops extend over the buffer memory of two ports: In case of (VC:input,Link:input) this is the output port and the input port of the next downstream switch. In case of (VC:output,Link:output) this is the input and the output port of this switch. In both cases there is a total buffer requirement of twice the amount of eq. 7, which is a quite a waste of memory.

The case (Link:output, **fig. 8**) has the severe problem that if (only if) buffer sharing is used (to reduce M^i_{max} in the input port) there are buffer requirements of $\sum_{\forall l} l^u_{link=l}$ in the output port, because up to $l^u_{link=l}$ cells from *each* port l can be queued there.

One of the hardest problems with internal flow control is the internal handling of the link credits. The link flow control loop (eq. 2) is designed to prevent the input ports from buffer overflow. Therefore the forwarded cell counter f_l must count cells for the correct input port, even when it is implemented in the output port. In contrast, the downstream loop is naturally associated with one output port. When downstream flow control cells (new credits) arrive on this output,

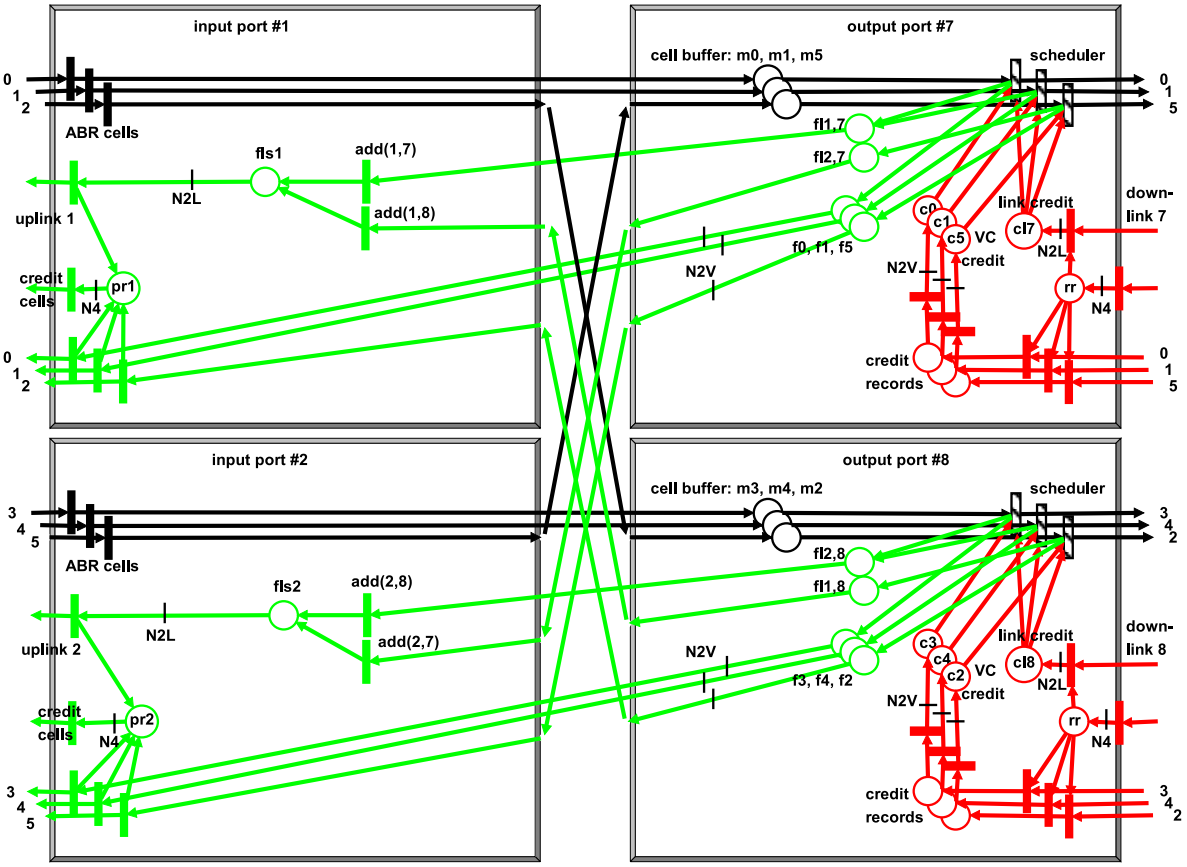

Figure 8: PN system for internal flow control method A (FC:output) [OQ]

port numbers	outputport *rightarrow*	7	8
inputport *downarrow*	notation	$V_7^o =$	$V_8^o =$
1	$V_1^i =$	0,1	2
2	$V_2^i =$	5	3,4

Table 4: connection numbers for example scenario

the state of this link changes. But if input ports handle the downstream FC test (TL), there is the problem how to dispatch (DFL) the credits to each input port so that each of them gets the share it will demand without having unused credits leftover at other ports. The following section provides solutions for these problems.

5.1 Petri-Net models

The most efficient methods are discussed here based on an example scenario utilizing two input (1&2) and output (7&8) ports and three connections through each. **Table. 4** lists the used connection numbers. For each method a PN system is given in which the four ports and their flow control scheme is visible. By analyzing the loops the general results in **table 2** and **3** have been obtained. They are independent of the architecture (OQ/VOQ). However, with knowledge about it only a few useful combinations remain:

Fig. 8 shows (VC:output,Link:output), where we assume an OQ architecture which doesn't queue cells in input ports ($M^i_{max} = 0$). Thus the total buffer requirements are only $l_{max} = \min(\sum_{\forall v} l^u_v, \sum_{\forall l} l^u_{link=l})$. For the construction of the link records there has to be a seperation of returned tokens for the corresponding input port i of a connection (places $fl_{i,o}$) and an addition of all those returned tokens that come into an input port i (places fls_i). If there is no buffer sharing and the architecture is OQ, this is the best solution. For VOQ this method isn't efficient due to its need for additional buffers of the same size in the input ports.

Fig. 9 shows (VC:input,Link:input), where the problem of dispatching the link backpressure to all input ports has been solved by maintaining one on/off state machine per output port o which reflects the availability of link credit in this output port o. The on-state of o is distributed to each input port i via the places $ON_{i,o}$, so that each input port keeps track of the the state of all output ports. Cells are only scheduled to output o if the place $ON_{i,o}$ has a token, which is achieved by the enabling arc construction. This method buffers all cells, for which the FC inequalities (1,2) are false, in the input port. In VOQ switches this is the only buffer loca-

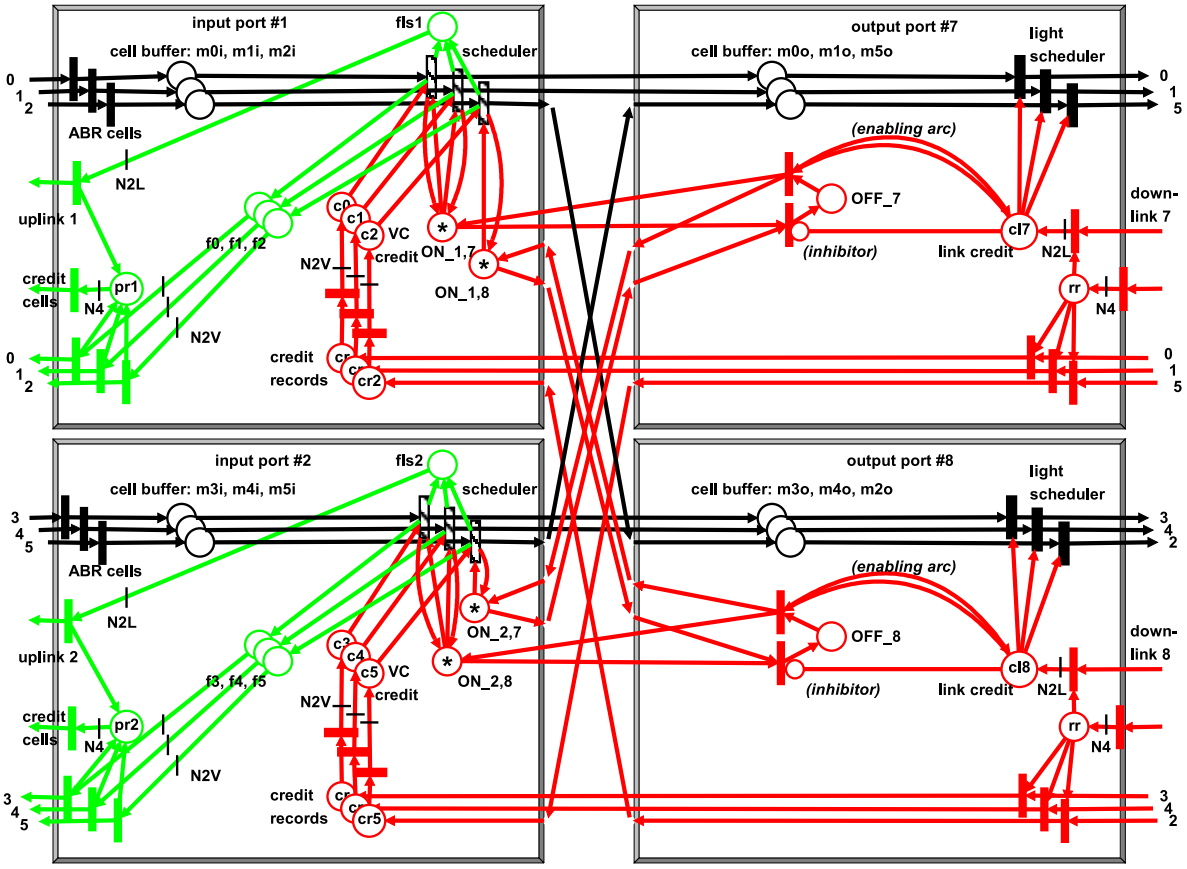

Figure 9: PN system for internal flow control method B (FC:input) [VOQ]

tion, so this offers an efficient method with minimum buffer requirements. There is the usual input buffer $M^i_{max} = \min(\sum_{v \in V^u_i} l^u_v, l^u_{link})$ and only a very small output buffer (potentially zero) just to compensate for the reaction time until an output link is stopped.

The method (VC:input,Link:input+output) shown in **fig. 10** differs from the previous by an internal link flow control that limits the output buffer usage to $l^x_{link=l}$, which only needs to store a few cells. The link backpressure is dispatched by the same method as above using on/off state machines. This method is suitable for both OQ and VOQ switches and requires only $l^x_{link=l}$ more buffer than the optimum. Since (VC:output,Link:output) is more efficient for OQ, this method is more attractive for VOQ switches with no or only a modest speedup.

In **fig. 11** (VC:input+output,Link:output) the internal backpressure is organized per connection, where for each connection a certain amount of buffer (l^x_v) is reserved. This method requires more total buffer than the previous and this increases with the number of connections, because it cannot be shared among connections. The buffering is input-dominant, so it is well suited for VOQ. In OQ switches this is less efficient than having only one buffer location in the output ports.

method	A	B	C	D
max rate	M	M	M	0
mean rate	$M/N2$	N/A	N/A	0

Table 5: message rate for port backpressure per timeslot and port

The models show that the location of the huge buffers (l_l) and the flow control test should be just before the bottleneck. For OQ this is the output port, for VOQ the input port. When secondary buffers must be avoided, method A is best suited for OQ and method B should be used for VOQ.

So far the efficiency of the variants have been studied in terms of buffer memory. Another question important for implementation is the bandwidth of the internal backpressure message streams from all output ports to all input ports. Each method transfers the forwarded notification for connections (VC) across ports. In the worst case ABR cells fill all links. Then the peak message rate (VC) per port is $M/timeslot$. Due to the quantization operation the mean rate (VC) is $M/N2/timeslot$. The methods differ only in the fre-

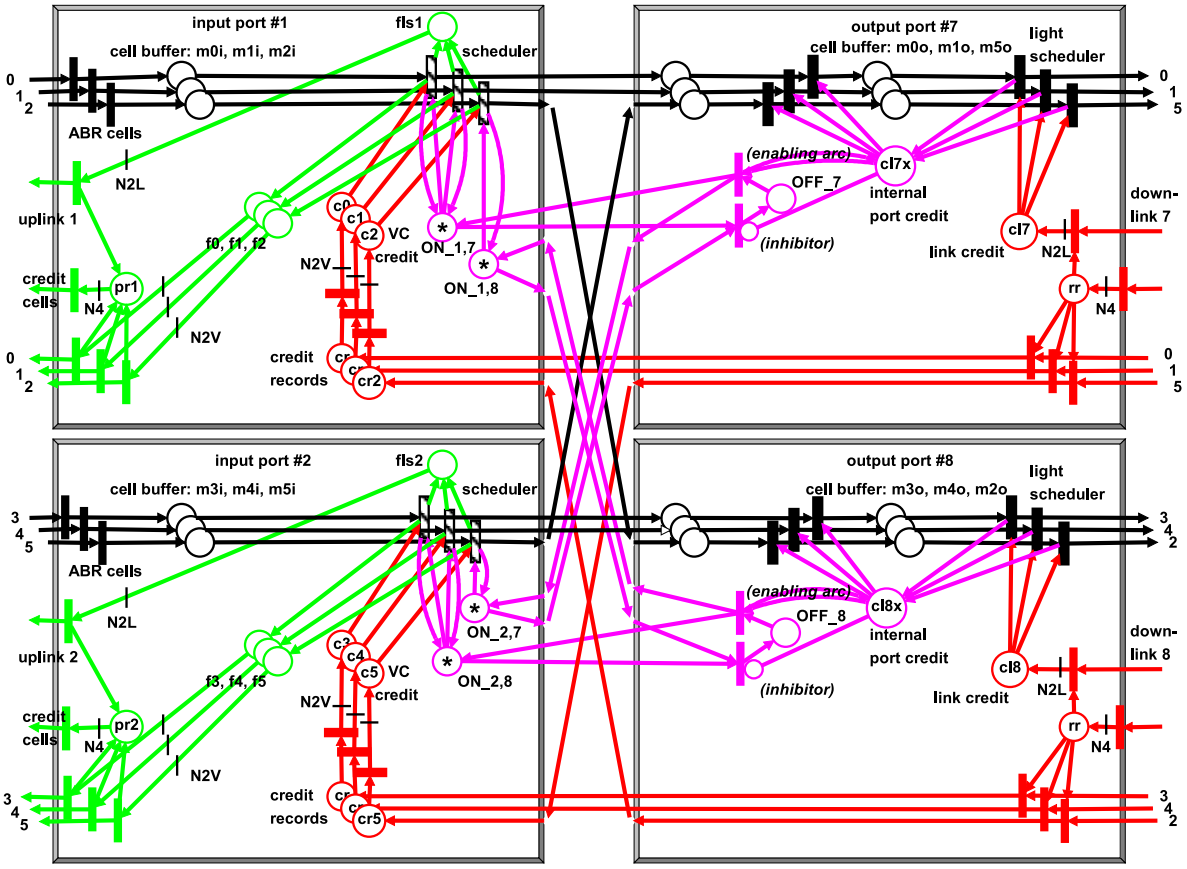

Figure 10: PN system for internal flow control method C (Link-FC:internal) [OQ+VOQ]

quency of link (port) backpressure. This is shown in **tab. 5**.

6 Conclusion

In this paper we have presented and compared several switch internal backpressure mechanisms to provide closed-loop flow control. This enables credit-based flow control to be used with new high-speed switching architectures, especially VOQ. The internal methods can also be applied within virtual nodes of all other flow control algorithms, e.g. explicit-rate. By using Petri net (PN) models, properties of stability and boundedness of memory have been studied to reveal pros and cons of the proposed alternatives. PN systems for the four best performing methods have been discussed with implications for the OQ and VOQ architecture. The external flow control behavior shows no difference with all of these methods, Simulation results with a black-box model can be used [7].

Open issues for future research are the support of point-to-multipoint connections and *MCR*-aware scheduling.

References

[1] R. Jain, "Congestion control and traffic management in ATM networks: recent advances and a survey," Feb. 1995. Computer Networks and ISDN Systems; available at ftp://netlab.ohio-state.edu/pub/jain/papers/cnis.ps.

[2] A. Mekkittikul and N. McKeown, "A Starvation-free Algorithm For Achieving 100% Throughput in an Input-Queued Switch," in *Proc. of the IEEE International Conference on Communication Networks*, 1996.

[3] N. McKeown, *Scheduling Algorithms for Input-Queued Cell Switches*. PhD thesis, UC Berkeley, 1995.

[4] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," *IEEE Transactions on Communications*, vol. 47, Aug 1999.

[5] A. Mekkittikul and N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches," *Proceedings of the IEEE INFOCOM*, 1998.

[6] H. Kung and R. Morris, "Credit-Based Flow Control for ATM Networks," *IEEE Network Magazine*, vol. 9, pp. 40–48, March/April 1995.

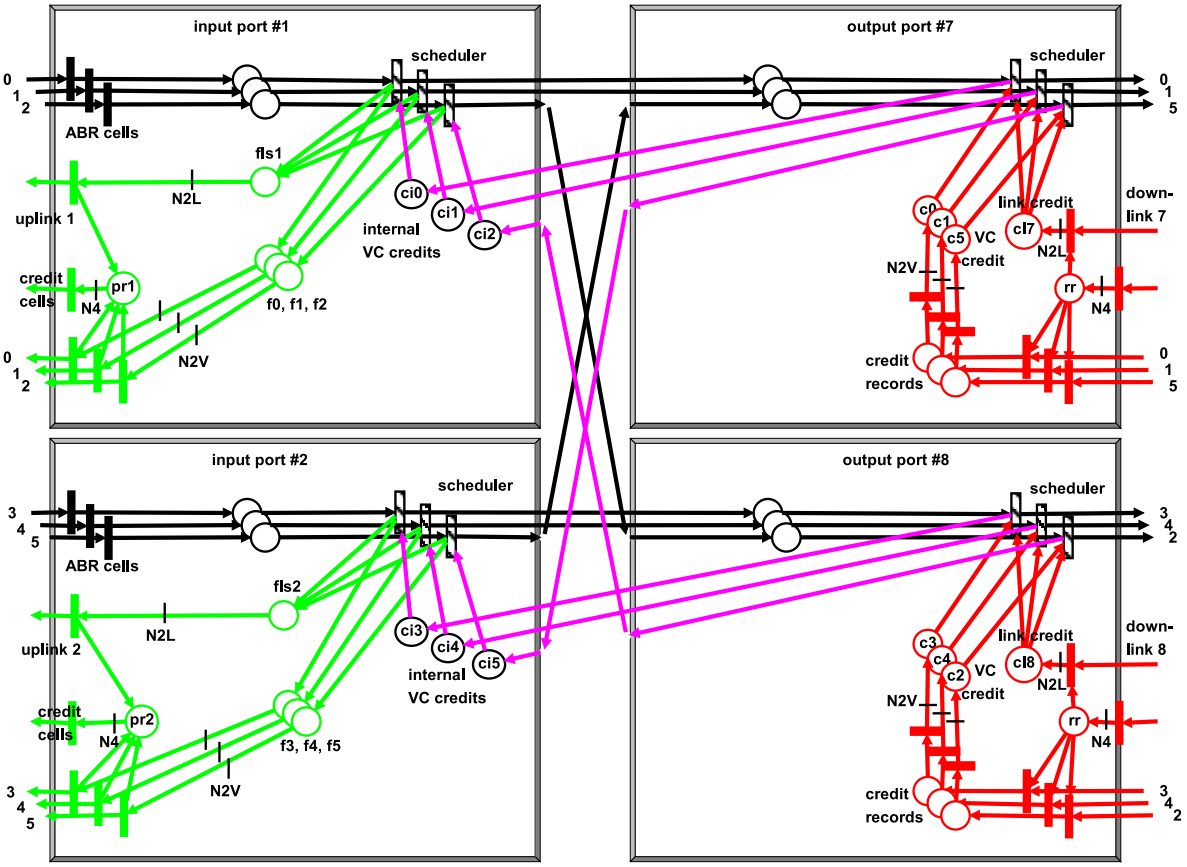

Figure 11: PN system for internal flow control method D (VC-FC:internal) [VOQ]

[7] R. Schoenen, G. Post, and A. Müller, "Analysis and Dimensioning of Credit-Based Flow Control for the ABR Service in ATM Networks," in *Proceedings of the IEEE GLOBECOM*, 1998. Vol.4 p.2399.

[8] T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, vol. 77, pp. 541–581, April 1989.

[9] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High Speed Switch Scheduling for Local Area Networks," *ACM Transactions on Computer Systems*, vol. 11, Nov 1993.

[10] Y. Tamir and H.-C. Chi, "Symmetric Cross Bar Arbiters for VLSI Communication Switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 1, pp. 13–27, 1993.

[11] R. Schoenen, G. Post, and G. Sander, "Weighted Arbitration Algorithms with Priorities for Input-Queued Switches with 100% Throughput," in *Proceedings of the IEEE Broadband Switching Systems*, 1999.

[12] R. Schoenen, G. Post, and G. Sander, "Prioritized Arbitration for Input-Queued Switches with 100% Throughput," in *Proc. of ATM Workshop'99*, 1999.

[13] N. McKeown, B. Prabhakar, and M. Zhu, "Matching Output Queueing with Combined Input and Output Queueing," in *35.th Annual Conf. on Communications, Control and Computing, Monticello, Illinois*, Oct 1997.

[14] M. . Bossart, S.-Y. Park, J. Lockwood, and S.-M. Kang, "ABR Architecture and Simulation for an Input-Buffered and Per-VC Queued ATM Switch," *Proceedings of the IEEE GLOBECOM*, vol. 1, p. 393, 1998. http://ipoint.vlsi.uiuc.edu/people/lockwood.

[15] The Flow Control Consortium, *Quantum Flow Control, Revision 2.0.5*, Mar. 1997. WWW: http://www.qfc.org/.

[16] ITU-T, *Controlled Transfer*, 1999. ITU-T Q7/13 section 6.X.

[17] M. Marsan, *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1996. ISBN 0-471-93059-8.

[18] R. Schoenen, V. Živojnović, and H. Meyr, "An Upper Bound of the Throughput of Multirate Multiprocessor Schedules," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (München), Apr. 1997.

[19] H. Kung and S. Wang, "Zero Queueing Flow Control and Applications," in *Proceedings of the IEEE INFOCOM*, 1998.