Extending the Birkhoff-von Neumann Switching Strategy to Multicast Switches

Jay Kumar Sundararajan, Supratim Deb, Muriel Médard *

Abstract. This paper extends the Birkhoff-von Neumann unicast switching strategy to the multicast case. Using a graph theoretic model we show that the rate region for a traffic pattern is precisely the stable set polytope of the pattern's 'conflict graph', in the no-fanout splitting case. Computing the offline schedule is equivalent to fractional weighted graph coloring which takes polynomial time for perfect graphs. For a general conflict graph, we show that deciding achievability of a given rate vector is NP-hard, but can be done in polynomial time for the case of moderate multicast load. The result naturally leads to an offline schedule.

1 Introduction

Online unicast scheduling in input-queued switches is a well-studied problem. McKeown et al. [1] gave a scheduling algorithm known as the maximum weighted matching (MWM) algorithm which achieves 100 % throughput for all admissible unicast arrival patterns. One major drawback of MWM-based approaches is that they do not provide cell delay guarantees. The Birkhoff-von Neumann (BVN) switch proposed by Chang et al. [2] addresses this issue. The BVN switch provides 100 % throughput for all non-uniform traffic, and also gives deterministic cell delay guarantees for certain types of traffic.

The basic idea of the BVN approach is as follows. Suppose the average arrival rate of packets is known for every input-output pair. This rate requirement matrix is decomposed into a convex combination of permutation matrices. Since a permutation matrix naturally corresponds to a switch configuration, this decomposition leads to an offline[1] schedule. In this paper, we study the extension of this approach to the case when there are multicast flows within the switch.

A fundamental question that arises while trying to extend the BVN strategy to multicast is that of how to split the *fanout* (the set of destinations) of each multicast flow. There are many options each requiring a different queuing policy:

1. **Copying:** Each input maintains one virtual output queue (VOQ) for every output. When a multicast cell arrives at an input, it is replicated as many

* The authors are with Laboratory for Information and Decision Systems, Massachusetts Institute of Technology. E-mail: {jaykumar, supratim, medard}@mit.edu

[1] In our work, 'online' means decide the switch configuration for each slot based on the current queue occupancies, whereas 'offline' methods assume knowledge of the average rates for each flow, and decompose the rate vector into a sequence of switch states.

times as its fanout size. One copy is added to the VOQ of each of the outputs in the fanout. This is equivalent to viewing the multicast as a collection of unicast flows.

2. **No-splitting:** This is the other extreme, where a multicast cell is sent to all outputs in its fanout in a single time slot. Each input must maintain a separate VOQ for every multicast flow. Moreover, the switch should have *intrinsic multicast capability*[2].
3. **Fanout-splitting:** Multiple copies of the multicast cell are generated and, in each slot, one copy is transferred to a subset of the fanout which has not already got the cell. The fanout is thus split partially. Again, the switch needs to have intrinsic multicast capability. This can be implemented in two ways:
   - **Static splitting:** Here, the multicast traffic pattern is assumed to be known before hand. All cells of a flow are split in the same manner (which is decided offline). This is like replacing the original multicast with a set of "split flows", for which further splitting is not allowed. Each input must maintain a separate VOQ for every split flow. When a multicast cell arrives at an input, it is replicated according to the predecided policy, and one copy is added to the VOQ of each of its split flows.
   - **Dynamic splitting:** For this strategy, each input maintains a VOQ for every subset of the fanout. When a multicast cell arrives at an input, it is transferred to some subset of its fanout and then, re-enqued into the VOQ corresponding to the remaining part of the fanout. The way in which a multicast flow's fanout is split can vary from cell to cell.

Dynamic splitting is the least constrained approach and subsumes the other options. It is known that, in the online case, dynamic fanout-splitting gives better throughput than no-splitting [3]. However, this benefit comes at a cost. Since the way flows are split is not known beforehand, the part of the fanout that remains to be served at some point of time, could be any arbitrary subset of the original fanout. Therefore, each input has to maintain a separate VOQ for every possible subset of the fanout, resulting in an exponential number of queues, even if the traffic pattern is known beforehand. In contrast, *the static splitting approach requires a much smaller number of queues.* At each input, one VOQ for every split flow is enough.

Earlier work on multicast switching has focused on online algorithms when dynamic splitting is allowed. Marsan et al. [4] defined the optimal online multicast scheduling algorithm, and defined the capacity region, but did not give an explicit characterization.

In this paper, we present a graph theoretic model for the problem, and use it to derive the rate region of the no-splitting case. We also present an algorithm to decide the achievability of a given set of rates, and find an offline schedule, in the case of moderate multicast load.

[2] The ability to simultaneously transfer a cell to multiple outputs using simultaneous switching paths

2 Rate Region and Offline Schedule

Definition 1 (Conflict graph). *The conflict graph $G = (V, E)$ for a given traffic pattern is defined thus:*

V = *set of all flows to be served*

$E = \{(v_i, v_j) | \textit{flows } i, j \textit{ cannot coexist within a valid switch configuration}\}$.

The conflict graph brings out the connection constraints in the switch. A valid switch configuration consists of a set of flows that can co-exist, which corresponds to a set of vertices no two of which are connected - in other words a *stable set*. A convex combination of stable sets gives a schedule where the flows in a particular stable set are served for a fraction of time equal to the coefficient in the combination. This leads to the following result. (The proof of the theorems in this paper are not given here for want of space, and can be found in [6]).

Theorem 1. *The rate region for the multicast case with no-splitting is precisely the stable set polytope of the conflict graph.* □

Next, we address the problem of computing the offline schedule in a general case. This approach gives an efficient algorithm for the case of perfect conflict graphs. [6] gives several examples of traffic flows whose conflict graph is perfect.

Theorem 2. *The problem of computing the decomposition of the given rate requirements into switch connection states (stable sets) is precisely the problem of* fractional weighted graph coloring *of the conflict graph, with the weights chosen as the required rates for the flows.* □

3 Deciding Achievability

Theorem 3. *The problem of deciding whether a given rate requirement vector is achievable using the no-splitting strategy is NP-hard.* □

The basic idea of the proof is to map the decidability question to the membership problem over the stable set polytope of a general graph. We next show that the hardness result can be extended to the case when dynamic fanout splitting is allowed. For details of the proof, please refer to [6].

Theorem 4. *The problem of deciding whether a given rate vector is within the rate region of the no-splitting case can be reduced to an instance of the corresponding problem in the fanout-splitting case.* □

Corollary 1. *The problem of deciding achievability in a multicast switch when fanout-splitting is allowed, is NP-hard.* □

We now present an algorithm to decide the achievability of a given rate vector, with no fanout-splitting. The algorithm runs in time polynomial in the switch size (N) if the number of multicasts is $O(logN)$(see [6] for details). This algorithm naturally gives a schedule to achieve the rates in a stable manner.

Algorithm 1 DECIDER:
INPUT: A rate requirement vector $\mathbf{r_o}$*; a traffic pattern in an* $N \times N$ *switch, with* k *multicasts and all possible unicasts.*
OUTPUT: Is $\mathbf{r_o}$ *within the achievable region corresponding to the traffic pattern, in the no-splitting case? If yes, give a schedule to achieve it in a stable manner.*

1. *Let the multicasts be* $M_1, M_2, \ldots, M_k$*. Let* $A \subseteq [k]$*. Let* R_A *be the rate region for the traffic pattern with the condition that the multicasts* $\{M_i | i \in A\}$ *are all always being served. Compute* R_A *for all possible subsets* A *of* $[k]$*.*
2. *Verify whether* $\mathbf{r_o}$ *lies in the convex hull of all the* R_A*'s. The answer to this question is the output.*
   Let $\mathbf{B_j x} \leq \mathbf{c_j}, j = 1, 2, \ldots, J$ *be the set of convex regions representing* R_A*. Verifying whether a point* $\mathbf{r}$ *is in the convex hull of these regions is equivalent to verifying whether this linear program in the variables* $\mathbf{y_j}$ *and* ϕ_j *is feasible:*

$$\sum_{j=1}^{J} \phi_j = 1; \quad \phi_j \geq 0; \quad \mathbf{r} = \sum_{j=1}^{J} \mathbf{y_j}; \quad \mathbf{B_j y_j} \leq \phi_j \mathbf{c_j} \quad \forall j = 1 \text{ to } J. \quad (1)$$

4 Conclusions

In this paper, we have investigated offline multicast switch scheduling using a graph theoretic formulation of the problem in terms of conflict graphs. We have shown that the rate region for a given traffic pattern is the stable set polytope of the conflict graph. We have provided an algorithm to decide the achievability of a rate vector and find the offline schedule for a moderate number of multicasts, in the no splitting case. Note that static splitting can be seen as an instance of no-splitting, with the set of flows chosen as the split flows. So, all results derived for no-splitting strategy in this paper hold for the static splitting strategy, which is very attractive as it requires a much smaller number of queues than dynamic splitting. The *i*-SLIP [5] unicast algorithm can be extended to multicast using the conflict graph formulation presented here. This is explained in detail in [6].

References

1. N. McKeown, V. Anantharam, J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch", *Proc. of IEEE INFOCOM '96*, pp. 296-302.
2. C.S. Chang, W.J. Chen and H.Y. Huang, "Birkhoff-von Neumann input buffered crossbar switches," *Proceedings of IEEE INFOCOM 2000*, pp. 1614-1623, Tel-Aviv, Israel, March 2000.
3. N. McKeown, "A Fast Switched Backplane for a Gigabit Switched Router", *Business Comm. Review*, Vol. 27, No. 12, Dec. 1997
4. M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri: "Multicast traffic in input-queued switches: optimal scheduling and maximum throughput." *IEEE/ACM Trans. on Networking* Vol. 11, No. 3, June 2003, pp. 465-477.
5. N. McKeown, "The i-SLIP scheduling algorithm for input-queued switches", *IEEE/ACM Transactions on Networking*, Vol. 7, no. 2, Apr. 1999.
6. Jay Kumar Sundararajan, "Extending the Birkhoff-von Neumann Switching Strategy to Multicast Switching", S.M. Thesis, MIT, January 2005