Probability in the Engineering and Informational Sciences, **14**, 2000, 327–334. Printed in the U.S.A.

SCHEDULING OF AN INPUT-QUEUED SWITCH TO ACHIEVE MAXIMAL THROUGHPUT

EITAN ALTMAN AND ZHEN LIU
INRIA
Centre Sophia Antipolis
2004 Route des Lucioles
B.P. 93
06902 Sophia-Antipolis, France

RHONDA RIGHTER*
Department of Operations Management and Information Systems
Santa Clara University
Santa Clara, California 95053

Achieving high throughput in input-queued switches has been found to be difficult, especially when traffic is nonuniform in the sense that different inputs have very different cell generation rates. We show that for general arrival processes, 100% throughput can be achieved with a simple algorithm that is very easy to implement.

We consider a switch in which in each time slot, at most one cell may be transmitted from each input, and at most one cell may be received at each output. Cells that are destined for output j arrive at input i according to a stationary and ergodic process, and arrivals are queued at the input. The problem is to decide which inputs are to transmit to which outputs in each time slot in order to maximize throughput. Necessary conditions for stability are that the total arrival rate to each input must be less than 1, and the total arrival rate destined to each output must be less than 1. We propose a simple scheduling algorithm and show that with this algorithm the necessary conditions for stability are also sufficient.

1. INTRODUCTION

In input-queued switches for such networks as ATM networks, cells of a fixed length are to be switched from one of several inputs to one of several outputs. In each time

*This work was done while this author was visiting INRIA.

slot, at most one cell can be transmitted from each input and at most one cell can be received by each output. It has been shown that if each input has a single queue and the queues are served according to the FIFO discipline, the throughput is only about 58% when traffic is independent and uniform [16]. Part of the reason for this is the HOL (head-of-line) blocking that occurs when there is only one queue at each input. In this case, a later cell destined for a different output can be blocked by the HOL cell if the HOL cell is destined for an output that is already receiving a cell from a different input. Many scheduling algorithms have been proposed that maintain separate virtual queues for each output at each input by permitting the server to access all buffer positions, not just the first [3,15,22–26]. Simulations have shown that such heuristic algorithms perform well when traffic is independent and uniform, but they do not perform as well for nonuniform traffic [22,23]. These heuristics are generally based on approximating the maximum size matching in each time slot; that is, they try to maximize the number of input–output connections that have nonzero queues and therefore maximize the number of cells switched in each time slot. It is possible to achieve maximal throughput on the output links when cells are queued at the outputs, but this requires a faster switch that can transmit multiple cells from each input and to each output in each time slot.

Until the recent article by McKeown et al. [21], it was unknown whether it was possible to guarantee 100% throughput for input-queued switches, even with separate queues for each output at each input. They showed, using a quadratic Lyapunov function, that 100% throughput for nonuniform (as well as uniform) traffic is, in fact, achievable with separate queues by using a maximum weight bipartite-matching algorithm, where the weights are the queue lengths. By 100% throughput, we mean that the system is stable as long as the aggregate arrival rate at each input and for each output is less than the capacity of the switch for each input and each output, which is 1. As McKeown et al. note, their result is theoretical because their algorithm is not practical for implementation. It requires an $O(\overline{N}^3 \log \overline{N})$ computational cost in each time slot, where $\overline{N}$ is the maximum of M (the number of inputs) and N (the number of outputs). Also, they assumed that at most one arrival occurs at each input in each time slot, so an input to the switch cannot be shared by multiple users. Moreover, their proof requires that the arrivals form mutually independent Bernoulli processes. They also show that a simpler maximum size matching algorithm is not stable in general.

We show, for arbitrary marginally stationary and ergodic arrival processes, that when arrival rates are known, 100% throughput can be achieved with a static time division multiplexing scheme that requires an $O(MN + \underline{N}^4)$ computation to be performed only once, and off-line, where $\underline{N}$ is the minimum of M and N. If arrival rates are unknown, we show that an adaptive version of this scheme, where such computations are performed periodically, also achieves 100% throughput. The results hold under very mild statistical assumptions on the arrival processes. We only assume that they are marginally stationary and ergodic. They may be mutually dependent.

The solutions we propose are related to the TDM (time division multiplexing) switching literature (e.g., [12,13], and references therein, and [5,7]). These references deal, however, with a deterministic setup, in which a fixed and known amount of traffic is to be sent from each given input to each given output; the problem is then to minimize the time it takes to finish the transmission from all input to all output ports. In solving our stochastic problem, we shall use some ideas from related deterministic problems [9].

Our scheme can be inexpensively modified to improve its performance by making it more responsive to the current load, without sacrificing its stability properties.

2. THE MODEL

Before presenting the results, we define the model under consideration. We analyze an $M \times N$ switch with M inputs and N outputs. We assume that there are separate input queues for each output, so we do not have HOL blocking. More precisely, each input is associated with N queues, one for each output. We denote by queue ij the queue for cells arriving to input i and destined for output j. We consider a slotted queueing model in which in each time slot, at most one cell can be transmitted from each of the M inputs, and at most one cell can be received by each of the N outputs.

For all $1 \le i \le M$ and $1 \le j \le N$, let $A_{ij}(n)$ be the number of cells that arrive at queue ij in time slot n. We assume that for each pair ij, the arrival process $\{A_{ij}\}_n$ is stationary and ergodic (cf. [4]) with rate λ_{ij} (i.e., the average number of cells arriving in each time slot). The arrival processes (for different pairs ij) may be mutually dependent.

Let

$$m(\lambda) := \max\left\{ \max_{1\le i\le M} \sum_{j=1}^{N} \lambda_{ij} ; \max_{1\le j\le N} \sum_{i=1}^{M} \lambda_{ij} \right\}$$

be the maximum of the total arrival rates for any input and any output, where $\lambda = (\lambda_{ij})$ is the matrix of arrival rates. Because in each time slot, at most one cell can be transmitted from each of the M inputs, and at most one cell can be received by each of the N outputs in the switch, it is easy to see that a sufficient condition for the switch to be saturated (i.e., one of the queues grows infinitely) is $m(\lambda) > 1$.

We will present scheduling algorithms for which the switch is stable under the condition $m(\lambda) < 1$. Stability here means the convergence in probability of queue lengths to finite random variables. We shall consider both the case when input rates λ_{ij} are known to the scheduler and the case when they are unknown.

3. STATIC SCHEDULING

We first suppose that the input rates, λ_{ij}, $1 \le i \le M$, $1 \le j \le N$, are known. We will develop a time-division multiplexing scheme so that each queue ij receives an effective service rate that is greater than λ_{ij}; that is, over a cycle of length C slots, input i will be "connected" to output j for t_{ij} time slots with $\mu_{ij} := t_{ij}/C > \lambda_{ij}$ for all i and j. During these slots, cells from input i will be transmitted to output j as long as queue

ij is nonempty. If queue ij is empty during one of its t_{ij} slots, nothing will be transmitted from queue i or to queue j during that slot.

Once we show that it is possible to construct such a cyclic schedule, then it immediately follows that the system will be stable because we have essentially decoupled our $M \times N$ queues into individual queues, each with a service rate greater than its arrival rate. Indeed, using Loynes' scheme [4,20], one can show that the stochastic process describing the number of cells at any queue ij at the beginning of the cycles converges in probability to a random variable that is finite with probability 1. Observe that using renovation theory [2,6], one can show a slightly stronger result: For each pair ij, $i = 1, \dots, M$, $j = 1, \dots, N$, the queue length process, $\{Q_{ij}(n);\ n = 1,2,\dots\}$, couples with a stationary ergodic regime within a time that is almost surely finite, where $Q_{ij}(n)$ is the number of cells waiting to be transmitted from input i to output j in time slot n. (The stationarity and ergodicity are with respect to a C-step shift, where C is some integer; see [2, Sect. 6].)

Let $\bar{N} = \max(M,N)$, let $\epsilon = \epsilon(\lambda) = (1 - m(\lambda))/\bar{N}$, and consider the most significant digit of ϵ; that is, let $k = k(\lambda)$, be such that $1/10^k \le \epsilon < 1/10^{k-1}$. Let $\delta = \delta(\lambda) = 1/10^{k+1}$ and let $C = 1/\delta$. For each λ_{ij}, let μ_{ij} be obtained by rounding λ_{ij} up to the nearest multiple of δ and then adding δ. Finally, let $t_{ij} = C\mu_{ij}$.

For example, suppose $N = M = 2$ and $\lambda_{ij} = 0.48$ for $i = 1$, 2 and $j = 1,2$. Then, $m(\lambda) = 0.96$, $\epsilon = 0.02$, $k = 2$, $\delta = 0.001$, $C = 1000$, $\mu_{ij} = 0.481$, and $t_{ij} = 481$, for $i =$ 1, 2 and $j = 1,2$. (A cycle of 1000 slots takes less than half a millisecond to transmit on the AN2 system of Digital's Systems Research Center [3].) Note that for all i and j, $\lambda_{ij} < \mu_{ij}$ and

$$\bar{C} := \max\left\{ \max_{1\le i\le M} \sum_{j=1}^{N} t_{ij};\ \max_{1\le j\le N} \sum_{i=1}^{M} t_{ij} \right\} < C.$$

The latter follows because

$$\bar{\mu} := \max\left\{ \max_{1\le i\le M} \sum_{j=1}^{N} \mu_{ij};\ \max_{1\le j\le N} \sum_{i=1}^{M} \mu_{ij} \right\} \le m(\lambda) + 2\bar{N}\delta < m(\lambda) + \bar{N}\epsilon = 1. \qquad \textbf{(1)}$$

Now, we must construct a schedule for "connecting" each input i with each output j for t_{ij} time slots during each cycle of length C. The constraints on our schedule are that while input i is connected to output j, input i can be connected to no other output and output j can be connected to no other input. Thus, our scheduling problem reduces to minimizing the makespan (the time it takes to complete all processing) for a preemptive open shop. This is a classical machine-scheduling problem in which there are M jobs and N machines, the processing time of job i on machine j is t_{ij}, jobs may be processed by machines in any order and may be preempted at any time, a job can be processed by at most one machine at a time, and a machine can process at most one job at a time. This problem can be solved in polynomial time, and the optimal makespan equals $\bar{C}$ [9]. In [10], an algorithm of time complexity $O(MN + \underline{N}^4)$ was provided for this problem, where $\underline{N} = \min\{M,N\}$. Examination of the algorithm [17] shows that when processing times are integer-valued, preemption will only occur at integer times, so the solution conforms to our slotted system.

The makespan result for the open-shop problem is similar to the Slepian–Duguid theorem for the nonblocking rearrangeability of switches in circuit-switched networks [12]. Anderson et al. use this approach for bandwidth allocation for CBR (constant bit rate) traffic [3]. Also, many algorithms have been developed in the TDM switching literature for finding optimal assignments of time slots to achieve a makespan, or frame length, of $\bar{C}$ given the t_{ij}'s (e.g., [5,13]). In [7], a parallel algorithm was developed. One of the major concerns in the TDM switching literature is to minimize the number of switching modes among solutions achieving minimum makespan.

Because the minimal makespan is of length $\bar{C}$, in each cycle of length C we implement the schedule obtained by solving the open-shop scheduling problem for the first $\bar{C}$ time slots, and we idle the switch for the remaining $C - \bar{C}$ time slots. Of course, we would do even better by not idling and letting our cycle length equal $\bar{C}$, but the longer cycle will be useful for the next section. This policy is easily implementable and will be stable as long as $m(\lambda) < 1$ (i.e., we can achieve 100% throughput).

THEOREM 3.1: *Assume that the arrival process* $\{A_{ij}\}_n$ *is stationary and ergodic for each pair of* (i, j), $1 \le i \le M$, $1 \le j \le N$. *If the arrival rates are known and if* $m(\lambda) < 1$, *then the switch is stable under the above-described static scheduling policy*.

The solution to the open-shop scheduling problem can be implemented in many ways, so we can choose a solution to minimize cell delays, for example. Indeed, the solution to the scheduling problem will give us a set of pairs (π_k, s_k), $k = 1,2,\ldots,$ where π_k is a permutation of $\{1,2,\ldots,\bar{N}\}$ specifying the connection between inputs and outputs [i.e., input i is connected to output $\pi_k(i)$] and s_k is the total number of slots where such a connection is used in a cycle. Note that there are $\bar{N} - \underline{N}$ dummy inputs or outputs. It is clear that the number of such pairs in the scheduling solution is bounded above by $\bar{N}!$. Actually, in the solution of [10], the number of preemptions is bounded above by $O(\underline{N}^3)$ and so is the number of such pairs. In the TDM switching literature, this number is bounded above by $O(MN)$ in many proposed algorithms.

A trivial implementation of the solution is to make connections according $\pi_1, \pi_2, \ldots$ for times $s_1, s_2, \ldots$ successively. Consider our earlier example with $N = M = 2$ and $\lambda_{ij} = 0.48$ for $i = 1, 2$ and $j = 1,2$, so $t_{ij} = 481$, for $i = 1, 2$ and $j = 1,2$. The (trivial) solution to the open-shop scheduling problem tells us that input i should be connected to output $i\,[\pi_1(i) = i]$, $i = 1,2$, for $s_1 = 481$ time slots, and input i should be connected to output j, $i = 1,2$, $j = 2,1[\pi_2(1) = 2$ and $\pi_2(2) = 1]$ for $s_2 = 481$ time slots, within a cycle of 1000 time slots. This solution gives two switching modes in the cycle and minimizes the number of switching modes. On the other hand, to minimize the delay, it is better to alternate the connections (the two permutations) in each time slot, assuming that the number of arrivals in each slot is independent and identically distributed.

Consider a general $N \times N$ switch with i.i.d. and uniform arrivals (so all arrival rates λ_{ij} are identical). Let $\gamma_1, \ldots, \gamma_N$ be the permutations on $(1,2,\ldots,N)$ such that $\gamma_k(i) = [(i+k-2) \bmod N] + 1$. It can be shown, using the techniques of [18], that in order to minimize mean cell delay, one can implement the cyclic scheduling policy which makes connection γ_k at all time slots $nN+k$, $n = 0,1,2,\ldots$, $k = 1,2,\ldots,N$. Indeed, such an implementation results in a cyclic service for all the output queues associated with any of the inputs.

For nonuniform arrivals, the optimization problem for minimizing delay is more complicated. In general, we should serve the input queues in the most "regular" way. For example, for the 2×2 switch, a good implementation can be obtained by applying techniques in the scheduling literature on optimal "splitting" [1,8,11,19]. For general switches, a reasonable policy is the "Golden ratio" policy [14]. Consider the following example which is adapted from [14]. Suppose $M = N = 3$, $\lambda_{11} = \lambda_{22} = \lambda_{33} = \frac{1}{2} - \epsilon$, $\lambda_{12} = \lambda_{23} = \lambda_{31} = \frac{3}{8} - \epsilon$, and $\lambda_{13} = \lambda_{21} = \lambda_{32} = \frac{1}{4} - \epsilon$, with $\varepsilon > 0$. Then, we can implement a cycle of length eight slots with $\gamma_1 = (1,2,3)$, $s_1 = 4$, $\gamma_2 = (2,3,1)$, $s_2 = 3$, and $\gamma_3 = (3,1,2)$, $s_3 = 1$, and the golden ratio schedule for each cycle of eight slots would be $\gamma_1\gamma_2\gamma_1\gamma_3\gamma_2\gamma_1\gamma_2\gamma_1$.

We can also improve our algorithm by combining it with existing heuristics, without sacrificing the stability property of our approach. Suppose for a given time slot, some of the scheduled connections have empty queues; that is, there exist inputs $i_1, i_2, \ldots i_K$, $K \leq \underline{N}$, such that there are no cells to transmit from i_k to $\pi(i_k)$ for $k = 1,\ldots,K$. Then, we can apply a heuristic to dynamically schedule those K inputs and outputs. Because this reallocation only removes a connection scheduled by our algorithm when there is nothing to transmit, the effective service rate for each queue is the same as it was in the original algorithm, so the system will still be stable when $m(\lambda) < 1$.

4. ADAPTIVE SCHEDULING

Now, let us suppose that the arrival rates are unknown. Let $A_{ij}(0,t)$ be the number of arrivals from input i destined to output j (i.e., the number of arrivals to queue ij) that have occurred by time t, and let $\hat{\lambda}_{ij}(t) := A_{ij}(0,t)/t$ be the estimated arrival rate for queue ij at time t. Let $\hat{\lambda}_t = (\lambda_{ij}(t))$ be the matrix of empirical arrival rates at time t. We assume that the estimate is updated infinitely often so that $\hat{\lambda}_t$ converges to λ with probability 1 (w.p.1). In particular,

$$t_0 := \inf\{t\colon m(\hat{\lambda}_t) < 1\}$$

is finite w.p.1. We assume that for each pair ij, the arrival point process $A_{ij}(0,t)$ is stationary and ergodic. The processes corresponding to different ij may again be mutually dependent.

Consider the following adaptive scheme:

1. Use an arbitrary scheduling policy before t_0 (e.g., the uniform scheduling policy γ of the last section or one of the existing heuristics).

2. Set $t := t_0$, $n = 1$.
3. Compute $\delta(\hat{\lambda}_t)$ and $C(\hat{\lambda}_t)$ as in the previous section. Also, compute $\mu_{ij}(\hat{\lambda}_t)$ in a way that is similar to that of the previous section: round up to the nearest multiple of δ and add *twice* δ. Set $t_{ij} = C\mu_{ij}$.
4. Use the scheduling policy described in the previous section [with respect to $t_{ij}(\hat{\lambda}_t)$] during the next $C(\hat{\lambda}_t)$slots.
5. (a) If for all pairs ij, $\hat{\lambda}_{ij}(t + nC(\hat{\lambda}_t)) \le \hat{\lambda}_{ij}(t) + \delta(\hat{\lambda}_t)$, then set $n := n + 1$ and repeat step 4.
   (b) Else if $m(\hat{\lambda}_{t+nC(\hat{\lambda}_t)}) \ge 1$, use an arbitrary scheduling policy until time $t_0 := \inf\{s > t + nC(\hat{\lambda}_t) : m(\hat{\lambda}_s) < 1\}$, and go to step 2.
   (c) If for some pair ij, $\hat{\lambda}_{ij}(t + nC(\hat{\lambda}_t)) > \hat{\lambda}_{ij}(t) + \delta(\hat{\lambda}_t)$ but $m(\hat{\lambda}_{t+nC(\hat{\lambda}_t)}) \le 1$, then set $t := t + nC(\hat{\lambda}_t)$ and $n = 1$, and go to step 3.

We now show that the above algorithm is stable whenever $m(\lambda) < 1$. Because $\hat{\lambda}_t$ converges to λ w.p.1, it follows that there exists some (random) time T which is finite w.p.1 and such that for all $t \ge T$, $m(\hat{\lambda}_t) < 1$, and for all pairs ij,

$$|\hat{\lambda}_{ij}(t) - \hat{\lambda}_{ij}(T)| < \delta(\hat{\lambda}_T). \tag{2}$$

Hence, after a time T that is finite w.p.1, steps 5(b) and 5(c) are never performed, and a fixed periodic schedule corresponding to $\hat{\lambda}_T$ with $m(\hat{\lambda}_T) < 1$ is used. Because $\mu_{ij} \ge \hat{\lambda}_{ij}(T) + 2\delta(\hat{\lambda}_T)$, step 2 implies that from time T onward,

$$\mu_{ij} \ge \lambda_{ij} + \delta(\hat{\lambda}_T) > \lambda_{ij}.$$

Moreover, the last inequality of (1) still holds:

$$\bar{\mu} := \max\left\{ \max_{1\le i\le M} \sum_{j=1}^{N} \mu_{ij}; \max_{1\le j\le N} \sum_{i=1}^{M} \mu_{ij} \right\} \le m(\lambda) + 3\bar{N}\delta < m(\lambda) + \bar{N}\epsilon = 1.$$

The discussion from the previous section then implies that the system is stable: For each i and j, there is a coupling convergence of the process of the number of ij cells in the system to a stationary ergodic process; that is, for every pair ij, the queue length process, $\{Q_{ij}(n);\, n = 1,2,\ldots\}$, couples with, or becomes identical to, a stationary ergodic regime within a time which is almost surely finite. To conclude, we obtain the following.

THEOREM 4.1: *Assume that the arrival process* $\{A_{ij}\}_n$ *is stationary and ergodic for each pair ij*, $1 \le i \le M$, $1 \le j \le N$. *If* $m(\lambda) < 1$, *then the switch is stable under the adaptive scheduling policy defined earlier.*

References

1. Altman, E., Gaujal, B., & Hordijk, A. (1997). Balanced sequences and optimal routing. INRIA Research Report No. 3180; *IEEE Transactions on Automatic Control*, in press.
2. Altman, E. & Hordijk, A. (1997). Applications of Borovkov's renovation theory to non-stationary stochastic recursive sequences and their control. *Advances of Applied Probability* 29: 388–413.

3. Anderson, T., Owicki, S., Saxe, J., & Thacker, C. (1993). High speed switch scheduling for local area networks. *ACM Transactions on Computer Systems* 11: 319–352.
4. Baccelli, F. & Bremaud, P. (1994). *Elements of queueing theory.* New York: Springer-Verlag.
5. Bongiovanni, G., Coppersmith, D., & Wong, C.K. (1981). An optimum time slot assignment algorithm for an SS/TDMA system with variable number of transponders. *IEEE Transactions on Communications* COM-29: 721–726.
6. Borovkov, A.A. & Foss, S.G. (1992). Stochastically recursive sequences and their generalizations. *Siberian Advances in Mathematics* 2: 16–81.
7. Chalasani, S. & Varma, A. (1993). Parallel algorithms for time-slot assignment in TDM hierarchical switching systems. *IEEE Transactions on Communications* COM-41: 1736–1747.
8. Gaujal, B. (1997). Optimal allocation sequences of two processes sharing a resource. *Journal of Discrete Event Dynamic Systems* 7: 327–354.
9. Gonzalez, T. & Sahni, S. (1976). Open shop scheduling to minimize finish time. *Journal of the Association of Computing Machinery* 23: 665–679.
10. Gonzalez, T. (1979). A note on open shop preemptive schedules. *IEEE Transactions on Computers* C-28: 782–786.
11. Hajek, B. (1985). Extremal splittings of point processes. *Mathematics of Operations Research* 10: 543–556.
12. Hui, J. (1990). *Switching and traffic theory for integrated broadband networks.* Deventer, The Netherlands: Kluwer Academic Press.
13. Inukai, T. (1979). An efficient SS/TDMA time slot assignment algorithm. *IEEE Transactions on Communications* COM-27: 1449–1455.
14. Itai, A. & Rosberg, Z. (1984). A golden ratio control policy for a multiple-access channel. *IEEE Transactions on Automatic Control* AC-29: 712–718.
15. Karol, M., Eng, K., & Obara, H. (1992). Improving the performance of input-queued ATM packet switches. *IEEE INFOCOM* 110–115.
16. Karol, M., Hluchyj, M., & Morgan, S. (1987). Input versus output queueing on a space division switch. *IEEE Transactions on Communications* COM-35: 1347–1356.
17. Lawler, E.L. & Labetoulle, J. (1978). On preemptive scheduling of unrelated parallel processors by linear programming. *Journal of the Associations of Computing Machinery* 25: 612–619.
18. Liu, Z., Nain, P., & Towsley, D. (1992). On optimal polling policies. *Queueing Systems* 11: 59–83.
19. Loeve, J.A. (1995). Markov decision chains with partial information. Ph.D. dissertation, University of Leiden, Leiden, The Netherlands.
20. Loynes, R.M. (1962). The stability of queues with non-independent inter-arrival and service times. *Proceedings of the Cambridge Philosophical Society* 58: 497–520.
21. McKeown, N., Anantharam, V., & Walrand, J. (1996). Achieving 100% throughput in an input-queued switch. *IEEE INFOCOM* 296–302.
22. McKeown, N., Walrand, J., & Varaiya, P. (1993). Scheduling cells in an input-queued switch. *IEE Electronics Letters* 2174–2175.
23. McKeown, N. (1995). Scheduling algorithms for input-queued cell switches. Ph.D. thesis, University of California, Berkeley.
24. Obara, H. (1991). Optimum architecture for input queueing ATM switches. *IEE Electronics Letters* 27: 555–557.
25. Obara, H. & Hamazumi, Y. (1992). Parallel contention resolution control for input queueing ATM switches. *IEE Electronics Letters* 28: 838–839.
26. Obara, H., Okamoto, S., & Hamazumi, Y. (1992). Input and output queueing ATM switch architecture with spatial and temporal slot reservation and control. *IEE Electronics Letters* 28: 22–24.