Parallel switching techniques for fixed-packet networks

Sanjoy Baruah[*]

(Extended Abstract)

§1 Introduction. Packet-switched, connection-oriented, virtual-circuit based networking technologies such as Asynchronous Transfer Mode (ATM) are likely to form the basis of future integrated services networks that aim to support a wide variety of applications, including multimedia teleconferencing, multimedia information retrieval, file-transfer, "regular" telephone services, image browsing, etc.

ATM networking technology is centered around the concept of switching 53-byte *cells*, where the cell is the basic unit of data transfer. One of the major reasons behind this design choice is that it is possible to build *parallel* switches, in which several of these small, fixed-size, cells are simultaneously processed by different switching elements and each switching element takes the same amount of time to complete its job. While much research has been performed on the issue of providing deterministic quality of service (QoS) guarantees to connections in ATM networks, none of this research, to our knowledge, exploints the parallelism capabilities of such networks. In part, this is due to the inherent intractability of most problems in deterministic multi-resource (as opposed to *uni*-resource) scheduling. *The purpose of this extended abstract is to very briefly describe our research efforts at designing* fair *parallel switching strategies for ATM (and similar) networks that are based upon the few known efficient algorithms for multi-resource scheduling.*

Particularly with respect to multimedia applications, the data-transfer requirements of an application may often be modelled as a data "stream" that is generated at a fairly regular rate, subject to occasional bursts. This model has been formalized into the concept of a *flow* [1, 2, 3, 4], which considers the traffic of a particular connection to be a sequence of packets or cells generated by the source of the connection: each packet belonging to a flow passes through the same sequence of switches along a path, established at connection-admission time, from the source to the destination in the network. A connection request specifies the rate at which it intends to generate data (i.e., it specifies its flow parameters), and the request is admitted by the network if and only if this flow would not overload the network and cause a consequent unacceptable degradation of service to other connections.

§2 Fairness. Consider a link between two switches that is of bandwidth B Mbps. Suppose that n connections $c_1, c_2, \ldots, c_n$ share this link, and that each connection c_i is characterized by a flow rate of $f(c_i)$ Mbps. The ratio $w(c_i) \stackrel{\text{def}}{=} f(c_i)/B$ denotes the fraction of the total bandwidth on the link that connection c_i requires (clearly, it is necessary that $\sum_{i=1}^{n} w(c_i) \leq 1$).

The "fairest" bandwidth scheme is one that, over any time interval $[t_x, t_y]$, is able to send $(t_y - t_x) \cdot f(c_i)$ Mb of data of connection c_i through this switch (provided, of course, that there

[*]The University of Vermont/ Department of Computer Science, Votey 351/ Burlington, VT 05405. Email: `sanjoy@cs.uvm.edu`. Supported in part by the National Science Foundation (Grant No. CCR-9596282).

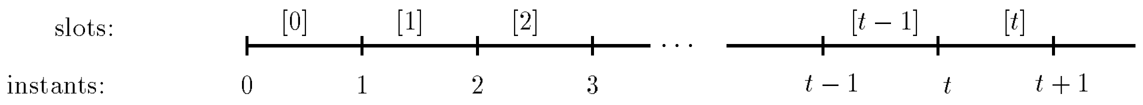

Figure 1: *Notation: Time instants and time slots*

is always some traffic waiting to be transmitted — i.e., that the connection is always *backlogged* during the interval $[t_x, t_y]$). However, achieving this would require that cells be infinitesimally small; since this is not the case, the "granularity" of fairness can be no smaller than the cell-size. More specifically, let a *time slot* denote the amount of time required to transmit one cell over the link — each time slot is equal to p/B sec, where p denotes the cell size in Mb (for ATM, $p = 424 \times 10^{-6}$). Assuming that time at each switch is measured in time slots numbered beginning with zero (Figure 1), the fairest bandwidth allocation scheme would be one that allocates at least $\lfloor t \cdot w(c_i) \rfloor$ slots out of any consecutive t slots to each connection c_i (provided, of course, that the connection is backlogged over this entire duration). Our attempt here is to achieve a slightly weaker form of fairness: suppose that there are no cells of connection c_i queued at the switch at the start of time slot $t_o - 1$, and that a cell arrives at the beginning of time slot t_o. In our version of fairness, *a bandwidth allocation scheme is fair if it allocates at least* $\lfloor (t_1 - t_o) \cdot w(c_i) \rfloor$ *slots out of the* $(t_1 - t_o)$ *slots numbered* $t_o, t_o + 1, \ldots, t_1 - 1$ *to connection* c_i (provided, once again, that the connection is backlogged during these slots).

§3 High-bandwidth connectivity. Modern networks, such as ATM-based ones, are typically constructed using optical fibres. One straightforward method of achieving greater bandwidth between two switches in such networks is to have several fibres in parallel connect the two switches — it costs significantly less to connect two locations with m fibres (typically, within the same cable) than it does to connect them with one fibre that has a bandwidth m times as high; more important, fast serial switching elements (i.e., switching elements that can handle a large number of cells per unit time) are very difficult to construct, and currently constitute the technological bottleneck to the implementation of higher-bandwidth networks. One method of having a high-bandwidth connection between two switches S1 and S2 is then to instead have several lower-bandwidth connections between them:

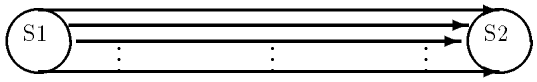

Figure 2: *An* $m \cdot B$ *bandwidth connection from switch S1 to switch S2 that is implemented by* m *fibres of bandwidth* B *each.*

At the beginning of each time slot, switch S1 would select up to m waiting cells in parallel and transmit them out on the m lines — through m different switching elements — to S2. These cells would arrive simultaneously at S2, and have to be processed in parallel; i.e., based upon the *Virtual Channel Indicator (VCI)* on each cell, the cell would be routed to the appropriate output queue in S2.

Assuming that each fibre has bandwidth B we could now, in principle, have n connections

$c_1, c_2, \ldots, c_n$ share the line between switches S1 and S2, provided that $\sum_{i=1}^{n} f(c_i) \leq m \cdot B$.

One of the major requirements of such an approach to increasing bandwidth is that, since the network offers a connection-oriented service, the relative ordering of the cells belonging to any particular connection be preserved. Specifically, if more than one cell of connection c_i is transmitted over the link during any time slot, it is absolutely essential that S2 be able to order these cells correctly before passing them on. This could be achieved, e.g., by adding a "cell-number" to each cell; however, such an approach would (i) slow down the switching process at S1 and S2 by increasing the amount of work that needs to be done at these switches, and (ii) be incompatible with the ATM standard as currently defined. Even if this were to be nevertheless done, sending several cells of the same connection during the same slot would require that these cells be compared and reordered at the receiving switch; these inherently sequential operations would reduce the amount of parallelism achieved in switch S2. All told, a better design decision would be to permit at most one cell of each connection to traverse the link during any time slot[1]. The problem of bandwidth allocation on the link S1–S2 thus reduces to the following problem:

The parallel switching problem: *Given n connections $C = \{c_1, c_2, \ldots, c_n\}$ and m parallel switches, with connection c_i needing to switch cells for* at most *a fraction $w(c_i)$ of the time slots, choose, for each time slot, a subset of C of size at most m of the connections that will be permitted to transmit cells during this time slot.*

Of course, we would like to be able to do this in as fair a manner as possible, where the idea of "fairness" should be closely related to the ideas discussed above. This is a *multi*-resource version of the problem of sharing a single resource in a fair manner; however, as far as we can tell, none of the single-resource fair queueing schemes suggested in the literature (such as Weighted Fair Queueing [5], Generalized Processor Sharing [6, 7], Fair Queueing Based on Start-times [8] or the schemes in [1, 4, 3]), nor the proportional-share schemes [9, 10, 11] generalize to this multi-resource problem. Detailing the exact reasons why this is so is beyond the scope of this extended abstract: we will, however, attempt to illustrate the problem by means of a simple example:

Example 1 Fair-queueing and proportional-share scheduling algorithms associate with each cell that is awaiting service at a switch an *eligible time* [9], which, loosely speaking, denotes the earliest time at which the packet is permitted to contend for the shared resource (in networks, this is the bandwidth). Consider, for example, a connection c_i with $w(c_i) = 2/3$. Assuming that connection c_i is continually backlogged starting at time 0, its first cell becomes eligible at time 0, the second at time 2, the third at time 3; in general, the j'th cell becomes eligible at time $\lceil 3(j-1)/2 \rceil$.

Consider now a situation where three connections c_1, c_2, and c_3, with $w(c_1) = w(c_2) = w(c_3) = 2/3$, share a parallel link composed of two fibres (i.e., with $m = 2$). Assume that all three connections are continually backlogged starting at time 0. At the start of the time-slot zero, all three connections are eligible; without loss of generality, assume that c_1 and c_2 get to send their cells during this slot. Since the next cells on these connections become eligible at time 2, connection c_3 is the only connection that has an eligible cell to send during time slot one; during this time slot, therefore, half the bandwidth has remained unused. Since $w(c_1) + w(c_2) + w(c_3) = 2$, this is clearly not acceptable.

■

[1]This design decision has two significant consequences: One, that no individual connection with a bandwidth requirement greater than B can be admitted over this link, despite the fact that the total link capacity is $m \cdot B$. Two, that we are no longer using a *work conserving* scheduling descipline — if less than m connections have cells waiting to use the link, then the link will not be used at full capacity despite the fact that there are cells that need to use it.

§4 Proportionate Progress and Pfairness. The notion of *proportionate progress*, and the associated concept of pfairness [12], deals with the following scheduling problem:

The multi-resource sharing problem: *Given n users $X = \{x_1, x_2, \ldots, x_n\}$ and m identical copies of a resource such that each resource must be allocated for each fixed (indivisible) quantum of time to a single user, and no user may use more than one copy of the resource during any time quantum, with user x_i needing to use the resource for* exactly *a fraction $w(x_i)$ of the time quanta, choose, for each time quantum, a subset of X of size at most m of the users that will be permitted to use the m resources during this time slot.*

It has been shown [12] that the m resources can be allocated in a *pfair* manner — i.e., in such a manner that, over the quanta numbered $0, 1, \ldots, t-1$, each user x_i will have received the resource for exactly $\lfloor w(x_i) \cdot t \rfloor$ or $\lceil w(x_i) \cdot t \rceil$ quanta, for all $t \in \mathbf{N}$; Algorithm PF [12] is a scheduling algorithm that determines the subset of users that obtain the m resources during each time quantum.

It should be evident that the multi-resource sharing problem is closely related to the parallel switching problem. However, the two problems have some major differences. The main difference — and the reason why Algorithm PF cannot be directly used to solve the parallel switching problem — lies in the fact that, while users are always waiting to use the resource, there may simply not be any cells of a connection queued up at the time that Algorithm PF would want to service that particular connection, but a cell arrives on this connection immediately after its "turn" has gone by. This is a consequence of the fact that, while everything about an instance of the multi-resource sharing problem — the number of users, their weights, the number of resources, etc. — is known beforehand, the parallel switching problem is inherently *on-line*, in that the exact times at which cells of a particular connection will arrive at a switch is not *a priori* known. Adapting multi-resource fair scheduling algorithms such as Algorithm PF (and the related Algorithm PD [13], which is more efficient from the run-time complexity point of view) to a dynamic, on-line environment while continuing to obtain high utilization of the available bandwidth, is the major algorithmic challenge in being able to design fair bandwidth allocation strategies for parallel switches.

§5 Other approaches. We conclude with a brief description of other promising approaches towards parallel switching of flow-based traffic that, based upon our preliminary studies, seem to merit further investigation. Of these, potentially the most rewarding is to explore the possibility of extending the single-resource fair-queueing strategies to the parallel-switching domain. While, as Example 1 illustrates, these are not likely to be optimal, the enormous amount of "legacy" research — the algorithms, implementations, and analyses — that has been performed with respect to these strategies may offset a minor loss of efficiency (bandwidth) or fairness. Another possible approach towards using the single-resource fairness results for parallel switching involves *partitioning* the connections among the various fibres at connection establishment time, and then having each connection's cells contend for bandwidth only on its associated fibre; once again, the downside to such an approach is low utilization of available bandwidth.

Recall that we had claimed that the "fairest" bandwidth allocation scheme would be one that allocates at least $\lfloor t \cdot w(c_i) \rfloor$ slots out of any consecutive t slots to each connection c_i. There seems to be a close relationship between this notion of extreme fairness and the concept of pinwheel scheduling [14]; further exploring this relationship may yield interesting results.

References

[1] L. Zhang. VirtualClock: A new traffic control algorithm for packet switching networks. In *Proceedings of ACM SIGCOMM'90*, pages 19–29, August 1990.

[2] H. Zhang and S. Keshav. Comparison of rate-based service disciplines. In *Proceedings of ACM SIGCOMM'91*, pages 113–121, August 1991.

[3] Amal El-Nahas, Khalil Ahmed, and Mohamed Gouda. Leaky bucket, refreshed bucket, and other flow admission criteria. In *Proceedings of the Fifth International Conference on Computer communications and Networks*, Rockville, MD, October 1996.

[4] Pawan Goyal, Harrick Vin, and Haichen Cheng. Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks. Technical Report TR-96-02, Department of Computer Sciences, University of Texas at Austin, 1996.

[5] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking Research and Experience*, pages 3–26, September 1990.

[6] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[7] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, April 1994.

[8] A. Greenberg and N. Madras. How fair is fair queueing? *Journal of the ACM*, 39(3):568–598, 1992.

[9] I. Stoica, H. Abdel-Wahab, K. Jeffay, J. Gherke, G. Plaxton, and S. Baruah. A proportional share resource allocation algorithm for real-time, time-shared systems. In *Proceedings of the Real-Time Systems Symposium*, Washington, DC, December 1996.

[10] I. Stoica and H. Abdel-Wahab. A new approach to implement proportional share resource allocation. Technical Report TR–95–05, Department of Computer Science, Old Dominion University, 1995.

[11] I. Stoica and H. Abdel-Wahab. Earliest eligible virtual deadline first: A flexible and accurate mechanism for proportional share resource allocation. Technical Report TR–95–22, Department of Computer Science, Old Dominion University, 1995.

[12] S. Baruah, N. Cohen, G. Plaxton, and D. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, June 1996. Extended Abstract in the Proceedings of the 1993 ACM Annual Symposium on the Theory of Computing.

[13] Sanjoy Baruah, J. Gehrke, and G. Plaxton. Fast scheduling of periodic tasks on multiple resources. In *Proceedings of the Ninth International Parallel Processing Symposium*, April 1995. Extended version available via anonymous ftp from `ftp.cs.utexas.edu`, as Tech Report TR–95–02.

[14] R. Holte, A. Mok, L. Rosier, I. Tulchinsky, and D. Varvel. The pinwheel: A real-time scheduling problem. In *Proceedings of the 22nd Hawaii International Conference on System Science*, pages 693–702, Kailua-Kona, January 1989.