Packet Routing via Min-Cost Circuit Routing

Baruch Awerbuch[*] *Yossi Azar*[†] *Amos Fiat*[‡]

Abstract

In this paper we initiate the study of competitive on-line packet routing algorithms. At any time, any network node may initiate sending a packet to another node. Our goal is to route these packets through the network, while simultaneously minimizing link bandwidth, buffer usage, and the average delay of a packet. We give efficient centralized on-line packet routing algorithms in this setting. These algorithms achieve a constant competitive ratio with respect to the average delay while increasing the link bandwidth by no more than a logarithmic factor.

To obtain our packet routing results, we introduce competitive algorithms for a new problem called min-cost load circuit routing. Here, the goal is to create on-line virtual circuits in a graph, while trying to simultaneously minimize link bandwidth and (related) communication costs.

1. Introduction

In this paper we initiate the study of competitive on-line packet routing algorithms. We consider a network where at any point in time, any network node may initiate sending a packet to another network node. Our goal is to route these packets through the network, while simultaneously minimizing link bandwidth, buffer usage, and the sum (or average) delay of a packet.

We give efficient *centralized* on-line packet routing algorithms in this setting. The major fault with our algorithm is that we require centralized decision control. This is a realistic model only if the packet sizes are truly large and control messages can be ignored. However, we hope that this note is but a first step towards a much greater understanding of on-line packet routing.

To obtain our packet routing results, we introduce competitive algorithms for a new problem, interesting on its own right. This problem is of min-cost load circuit routing. Here, the goal is to create on-line virtual circuits in a graph, while trying to simultaneously minimize link bandwidth and (related) communication costs.

[*]Johns Hopkins University and Lab. for Computer Science, MIT. Supported by Air Force Contract TNDGAFOSR-86-0078, ARO contract DAAL03-86-K-0171, NSF contract 9114440-CCR, DARPA contract N00014-J-92-1799, and a special grant from IBM. E-Mail: baruch@theory.lcs.mit.edu.

[†]Department of Computer Science, Tel Aviv University, Tel Aviv, 69978, Israel. Phone:+972-3-6407442. Fax:+972-3-6409357. E-Mail: azar@math.tau.ac.il. Research supported in part by Alon Fellowship and by the Israel Science Foundation administered by the Israel Academy of Sciences.

[‡]Department of Computer Science, Tel-Aviv University, Israel. E-Mail: fiat@math.tau.ac.il. Research supported in part by the Israel Science Foundation administered by the Israel Academy of Sciences.

1.1. Packet routing models and results

Network. We are given a network $G = (V, E)$ with $|V| = n$ and $|E| = m$. Every network node v can hold up to $cap(v)$ packets in its buffer, every link e has some non-negative capacity $cap(e)$. At every unit of time, up to $cap(e)$ packets can traverse an edge e. All packets sent along a link arrive at the links endpoint in one unit of time.

Input. The input consists of triplets s_i, r_i, t_i, which represent the source, the receiver, and the time of generation of the i's packet. The input is generated in an on-line fashion, *i.e.*, input i is disclosed at time t_i.

Output. The *output* is scheduling of packets satisfying the capacity constraints at the network nodes and edges, and delivering every packet in some finite time. We consider on-line scheduling algorithms that have access to global information, i.e. algorithm knows at time t the state of all the network edge and network node buffers, and, of course, past input (triplets s_i, r_i, t_i with $t_i \leq t$).

Remarks In our model above packets that arrive and are immediately retransmitted are not counted in the buffer utilization at network nodes. However, an alternative model would be to count them, our algorithms are easily modified to fit this model as well giving the same results.

Complexity measures. Define the sum (average) of the delay of a sequence as the sum (average) of the delays over all packets in the sequence.

An on-line algorithm is called α-*load* competitive and β-*delay* competitive if it can schedule any off-line feasible sequence of packets in the network such that the capacity of each link and buffer is increased by a factor of α and the sum of the delays (or average delay) is increased by a factor of at most β.

Our packet routing algorithm follows from a straightforward reduction (Section 3) to a special case of the problem of on-line *minimum cost* circuit routing, where the cost of a link is equal to its capacity. The solution for the latter problem, which is interesting in its own right, is given in Section 2.

The packet routing algorithm as described in Sections 3 and 2 is $O(\log(nT))$ capacity-competitive and $O(1)$ delay-competitive. Here, n is the number of network nodes, and T is the maximum packet delay of off-line schedule. (See Theorems 3.2 and 3.3).

We also design an algorithm which is competitive with respect to the maximum delay instead of average delay assuming infinite buffer capacity at all network nodes. It turns out that the latter is much simpler problem and using simple methods we design an on-line algorithm which is $O(1)$ capacity-competitive and $O(1)$ delay-competitive (see Section 4).

Comparison with related work. Packet routing is one of the classical problems in the areas of networking [18, 11, 19, 17, 10, 21, 24, 8, 6, 3, 5] and parallel computing [12, 23, 9, 16, 14, 25, 15, 7, 20]. Most of this work focused on *distributed* routing algorithms, where the overhead of control messages is taken into account, i.e., is applicable to the case in which the packets transmitted are *small*.

The analytical efficiency, i.e. polylogarithmic gaps between upper and lower bounds on packet delay, has been achieved primarily for the special-purpose parallel architectures such as hypercubes, expanders, etc.; good survey of this work can be found in [13]. The exception is the seminal work of Leighton, Maggs and Rao, [16], which achieve logarithmic (and, in some cases, constant) competitiveness for the *maximum* delay in a *static setting*.

In contrast, in this work, we consider, for the first time, the more difficult problem of minimizing *average* delay in an on-line setting. We, however, ignore the overhead of control, assuming knowledge of global state.

1.2. Min-cost circuit routing models and results

Input: A graph $G = (V, E)$ with a capacity function $cap : E \rightarrow R^+$ and cost function $cost : E \rightarrow R^+$. We get a sequence of requests for connections, each is defined by a tuple (s_i, r_i, p_i), where $s_i, r_i \in V$ are the source/sink pair, and $p_i \in R^+$ corresponds to the required bandwidth.

Output: At the time of arrival of request for i'th connection, the required bandwidth has to be reserved along some path P_i from the source s_i to the sink r_i.

Performance: Regardless of whether connection i, is using edge e or not, we define *relative load* on e due to connection i as $p_{i,e} = p_i/cap(e)$. Let $\mathcal{P}$ and $\mathcal{P}^*$ be the set of paths associated with the connections by the on-line and the off-line algorithms, respectively. The *relative load* on edge e of the on-line algorithm is defined by:

$$\ell_e = \sum_{P_i \in \mathcal{P}: e \in P_i} p_{i,e},$$

and the cost of the scheduling of the on-line algorithm is defined as

$$\sum_{e \in E} cost(e)\ell_e \ .$$

The relative load on edge e of the off-line algorithm is defined similarly and denoted by ℓ_e^*. The cost of the scheduling of an off-line algorithm is defined by the equation above by replacing ℓ_e by ℓ_e^*.

Definition 1.1 *An on-line circuit routing algorithm is called* α capacity-competitive *and* β cost-competitive *if, for each input sequence served, it is absolutely impossible to have served that sequence with* both α*-smaller relative load and* β*-smaller cost.*

Equivalently, we call a sequence of requests *feasible* if there exists an off-line algorithm that allocated a path for each request without exceeding the capacity constraints. This allocation is called a *feasible* solution. A feasible solution is equivalent to the property that the relative load on each edge of at most 1. The optimal off-line cost is the minimum over of possible feasible solutions of the cost of the scheduling. An on-line algorithm is with α capacity-competitive and β cost-competitive if it can schedule any request sequence by increasing the capacity of each link by a factor α and achieving a cost which is at most β times the optimal.

Theorem 1.2 *For arbitrary network with arbitrary capacities and costs our min cost circuit routing on-line algorithm is* $O\left(\log(\sum_e cost(e))\right)$ *load-competitive and* $O(1)$ *cost-competitive.*

2. Min-cost circuit routing algorithm

We first discuss the case where all the costs are 1. For this case we show an algorithm which $O(\log n)$ load-competitive and $O(1)$ cost-competitive.

Define $a = 1 + \gamma$ for some positive constant $\gamma < 1$. Let $h_{i,e}$ be the relative load ℓ_e on edge e at the time of the arrival of the i'th request. The on-line routing algorithm works as follows. Request i is assigned to an $s_i - r_i$ path P_i which minimizes

$$W_i^P = \sum_{e \in P} (a^{h_{i,e}+p_{i,e}} - a^{h_{i,e}}).$$

We use the techniques of [1, 2] to show the following invariant.

Lemma 2.1 *The algorithm maintains the following inequality:*

$$\sum_{e \in E} a^{\ell_e}(1 - \gamma \ell_e^*) \leq m.$$

Proof: Consider a connection i that is assigned by the on-line algorithm to some path $P = P_i$. Let $P^* = P_i^*$ be the path that is assigned to this connection by the off-line algorithm. Note that since $a = 1 + \gamma$ we have that $\forall x \in [0, 1] : \quad (a^x - 1) \leq \gamma x$. Moreover, the fact that the off-line algorithm routes the ith request through P^* implies that for each $e \in P^*$ we have $0 \leq p_{i,e} \leq 1$, and hence the inequality above applies for $x = p_{i,e}$. This and the fact that the algorithm chooses the minimum weight path implies that

$$\begin{aligned} W_i^P &= \sum_{e \in P} (a^{h_{i,e}+p_{i,e}} - a^{h_{i,e}}) \\ &\leq \sum_{e \in P^*} (a^{\ell_e + p_{i,e}} - a^{\ell_e}) \\ &= \sum_{e \in P^*} a^{\ell_e}(a^{p_{i,e}} - 1) \\ &\leq \gamma \sum_{e \in P^*} a^{\ell_e} p_{i,e}. \end{aligned}$$

Summing over all connections, we get:

$$\sum_{P \in \mathcal{P}} \sum_{e \in P} (a^{h_{i,e}+p_{i,e}} - a^{h_{i,e}}) \leq \gamma \sum_{P^* \in \mathcal{P}^*} \sum_{e \in P^*} a^{\ell_e} p_{i,e}.$$

Exchanging the order of summation yields

$$\begin{aligned} \sum_{e \in E} \sum_{P \in \mathcal{P} | e \in P} (a^{h_{i,e}+p_{i,e}} &- a^{h_{i,e}}) \\ &\leq \gamma \sum_{e \in E} a^{\ell_e} \sum_{P^* \in \mathcal{P}^* | e \in P^*} p_{i,e} \\ &= \gamma \sum_{e \in E} a^{\ell_e} \ell_e^*. \end{aligned}$$

Clearly, the left hand-side is a telescopic sum for each edge e. Thus we conclude that

$$\sum_{e \in E} (a^{\ell_e} - 1) \leq \gamma \sum_{e \in E} a^{\ell_e} \ell_e^*.$$

Hence,

$$\sum_{e \in E} a^{\ell_e}(1 - \gamma \ell_e^*) \leq m.$$

∎

Theorem 2.2 *The route algorithm maintains a relative load of at most $O(\log n)$.*

Proof: Observe that $\gamma < 1$ by definition $\ell_e^* \leq 1$ since the normalized load of the off-line algorithm never exceeds 1. Thus, Lemma 2.1 then implies that

$$\sum_{e \in E} a^{\ell_e} \leq m/(1 - \gamma).$$

This, in turn implies that

$$\max_{e \in E} \ell_e \leq \log_a(m/(1 - \gamma)) = O(\log n).$$

∎

Theorem 2.3 *The route algorithm maintains the following*

$$\sum_{e \in E} \ell_e = O(\sum_{e \in E} \ell_e^*).$$

Proof: Lemma 2.1 states that

$$\sum_{e \in E} a^{\ell_e}(1 - \gamma \ell_e^*) \leq m,$$

where $a = 1 + \gamma$ and $0 < \gamma < 1$. Using the inequality $1 + \frac{\gamma}{2}\ell_e \leq (1 + \gamma)^{\ell_e}$ for any $0 < \gamma < 1$ and $\ell_e \geq 0$ we get

$$\sum_{e \in E} (1 + \frac{\gamma}{2}\ell_e)(1 - \gamma \ell_e^*) \leq m.$$

By opening the parenthesis we have

$$m - \gamma \sum_{e \in E} \ell_e^* + \frac{\gamma}{2} \sum_{e \in E} \ell_e (1 - \gamma \ell_e^*) \leq m,$$

or,

$$\sum_{e \in E} \ell_e (1 - \gamma \ell_e^*) \leq 2 \sum_{e \in E} \ell_e^*.$$

Recall that $\gamma < 1$ and $\ell_e^* \leq 1$ for each e. Thus

$$\sum_{e \in E} \ell_e \leq \frac{2}{1 - \gamma} \sum_{e \in E} \ell_e^*.$$

∎

We extend the above result to the case where not all the costs are 1. Clearly we can replace an edge with capacity $cap(e)$ and cost $cost(e)$ with a line of $cost(e)$ edges of capacity $cap(e)$ and cost 1. That increases the number of edges in the graph to $\sum_e cost(e)$. We conclude that

Theorem 2.4 *For arbitrary network with arbitrary capacities and costs the on-line algorithm is competitive with* $O\left(\log(\sum_e cost(e))\right)$*-load and* $O(1)$ *cost-competitive.*

We note that for the above algorithm is equivalent to the the following algorithm on the original graph. Request i is assigned to an $s_i - r_i$ path P_i which minimizes

$$W_i^P = \sum_{e \in P} cost(e)(a^{h_{i,e}+p_{i,e}} - a^{h_{i,e}}).$$

3. Reducing packet routing to on-line min-cost circuit routing

In this section we show how to reduce the packet routing problem in graph $G = (V, E)$ to the min-cost virtual paths routing in a directed graph L on size $O(nT')$ where T' is the total duration of the packet routing sequence. Then we will show how to replace T' with T the maximum delay of a packet by the off-line algorithm. We construct the following layered graph L. The graph has $T' + 1$ layers indexed by time $0 \leq t \leq T'$ and one additional layer, which we call the destination layer. Each layer has n vertices one for each node in G. We define the set of edges, their capacities and costs as follows. For an edge $e = (u, v) \in G$ we make directed edges of capacity $cap(e)$ between vertex u in layer $t-1$ to vertex v in layer t for all $1 \leq t \leq T'$. Also we have a directed edges of capacity $cap(u)$ between vertex u in layer $t - 1$ to vertex u in layer t for all $1 \leq t \leq T'$. Finally, there are directed edges between all copies of vertex u to the destination vertex u. These edges are ignored for the purpose of load or cost.

For a request for routing a packet from s_i to r_i which is created at time t we associated a request for a virtual path from vertex s_i in layer t to the destination vertex r_i. It is easy to see that there is one to one correspondence between the packet route in G to the path in L. Specifically, an edge in L from vertex u of layer $t - 1$ to vertex v of layer t corresponds to routing the packet on the link from u to v at the t'th unit time. Also an edge from u in layer $t - 1$ to u in t corresponds to keeping the packet in the buffer of node u.

Note that the length of the path from the source to destination (except the edge to the destination layer) corresponds to the delay of the packet. Thus the sum of the actual loads (not the relative load) corresponds to the sum of the delays. Recall that for this purpose we ignore the edges to the destination layer. For each edge e we associate a cost cap_e which is the ratio between the actual load to the relative load. Thus, the min-cost problem corresponds to the packet routing problems which minimizes the sum (or average) delay.

Using the result from the previous section we conclude

Corollary 3.1 *There is an on-line algorithm for packet routing for a network with equal capacities which is competitive with* $O(\log(nT'))$*-load and* $O(1)$ *delay.*

By standard methods (see [4]) one can replace the $O(\log(nT'))$ by $O(\log(nT))$ where T is the maximum delay of a packet. This is done by constructing a new layered graph of $2T+1$ layers indexed by $Ti \leq t \leq T(i+2)$ for all $i \geq 0$ to accommodate requests that were created between Ti to $T(i + 1)$. We apply the algorithm for each graph separately. Clearly, at any unit of time there are at most 2 such active layers. This results in increasing the bandwidth of each edge by a factor of 2. Also, it is not difficult to deal with the case that T is unknown in advance. Again one can use the methods of [4] and double T until it reaches its approximate value. Thus we conclude

Theorem 3.2 *There is an on-line algorithm for packet routing for a network with equal capacities which is competitive with* $O(\log(nT))$*-load and* $O(1)$ *delay.*

If the capacities are not the same using the results from the previous section to conclude

Theorem 3.3 *There is an on-line algorithm for packet routing for a network with arbitrary capacities which is competitive with* $O\left(\log(T \sum_e cap(e))\right)$*-load and* $O(1)$*-delay.*

4. Maximum delay packet routing

In this section we describe an on-line algorithm which is competitive with respect to the maximum delay instead of the average delay. We design an algorithm which is competitive with 2-load and 3-maximum delay given the value of the maximum delay and assuming infinite buffer capacity at network nodes. Then, using standard methods we can eliminate the need for knowing the maximum delay by increasing the on-line maximum delay by a constant factor.

Let T be the maximum delay. All the requests appeared between time $T(i - 1)$ and Ti are collected and scheduled between Ti and $T(i + 2)$. Clearly this can be done since all the requests could be schedule with maximum delay T and hence requests that were created in a range of T units of time can be scheduled optimally in $2T$ units of time. Hence the maximum delay is increased by a factor of at most 3 (the delays can be as large as $T(i+2) - T(i-1)$). The capacity required to achieve this scheduling increases by a factor of 2 since at any time there are at most 2 scheduling graphs. We note that the scheduling the requests at each T steps is NP-hard. Thus, if we are ready to use super-polynomial time algorithms it is possible to schedule the packets. If on the other hand we restrict ourselves to polynomial algorithms then we should use off-line approximation algorithms, such as [22], and increase the bandwidth by the approximation factor.

If T is unknown in advance then again we can use the doubling technique described in [4]. That increases the maximum delay by a constant factor. Thus we conclude

Theorem 4.1 *There is an on-line algorithm for packet routing for a network with arbitrary edge capacities and infinite node capacities which is $O(1)$ competitive with respect to load and simultanously $O(1)$ competitive with respect to the maximum delay.*

5. Conclusion and open problems

In this paper, we have presented competitive on-line centralized algorithms for for packet routing, by reducing them to the problem of min-cost circuit routing.

The obvious open question is whether it is possible to design *distributed* algorithms with poly-logarithmic competitive ratios.

References

[1] Jim Aspnes, Yossi Azar, Amos Fiat, Serge Plotkin, and Orli Waarts. On-line load balancing with applications to machine scheduling and virtual circuit routing. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 623–631, May 1993.

[2] Baruch Awerbuch, Yossi Azar, Serge Plotkin, and Orli Waarts. Competitive routing of virtual circuits with unknown duration. In *Proc. 5'th ACM-SIAM Symp. on Discrete Algorithms*, pages 321–327, 1994.

[3] Alok Aggarwal, Amotz Bar-Noy, Don Coppersmith, Rajiv Ramaswami, Baruch Schieber, and Madhu Sudan. Efficient routing and scheduling algorithms for optical networks. In *Proc. 5'th ACM-SIAM Symp. on Discrete Algorithms*, pages 412–423, 1994.

[4] Yossi Azar, B. Kalyanasundaram, Serge Plotkin, K. Pruhs, and Orli Waarts. On-line load balancing of temporary tasks. In *Proc. Workshop on Algorithms and Data Structures*, pages 119–130, August 1993.

[5] Baruch Awerbuch and Tom Leighton. Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks. In *Proc. 26th ACM Symp. on Theory of Computing*, May 1994.

[6] Dimitri P. Bertsekas. Optimal routing and flow control methods for communication networks. In A. Bensoussan and J. L. Lions, editors, *Proc. Fifth Intl. Conf. on Analysis and Optimization*, pages 615–643, New York, December 1982. Lecture Notes in Control and Information Sciences.

[7] Alan Borodin, Prabhakar Raghavan, Baruch Schieber, and Eli Upfal. How much can hardware help routing? (extended abstract). In *Proc. 25th ACM Symp. on Theory of Computing*, pages 573–582, 1993.

[8] Krishna Bala and Tom Stern. Multiple channel routing in a linear lightwave network. In *OFC*, 1991.

[9] William Dally and Chuck Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. on Commun.*, 36(5), May 1987.

[10] Robert G. Gallager. An optimal routing algorithm using distributed computation. *IEEE Trans. on Commun.*, 25:73–85, January 1977.

[11] J. M. Jaffe and F. M. Moss. A responsive distributed routing algorithm for computer networks. *IEEE Transactions on Communication*, COM-30(7):1758–1762, July 1982.

[12] Charles E. Leiserson. Fat-trees: Universal networks for hardware efficient supercomputing. *IEEE Trans. on Computers*, C-34(10):892–901, October 1985.

[13] Tom Leighton. *Introduction to Parallel Algorithms and Architectures*. Morgan-Kaufman, 1991.

[14] T. Leighton and B. Maggs. Expanders might be practical: Fast algorithms for routing around faults in multibutterflies. In *Proc. 30th IEEE Symp. on Found. of Comp. Science*, pages 384–389, October 1989.

[15] T. Leighton and B. Maggs. Fast algorithms for routing around faults in multibutterflies and randomly-wired splitter networks. *IEEE Trans. on Computers*, 41(5):1–10, May 1992.

[16] T. Leighton, B. Maggs, and S. Rao. Universal packet routing algorithms. In *Proc. 29th IEEE Symp. on Found. of Comp. Science*, pages 256–271. IEEE, October 1988.

[17] N.F. Maxemchuk. Routing in the manhattan street network. In *IEEE Trans. on Communications*, volume COM-35, pages 503–512, May 1987.

[18] J. M. McQuillan, G. Falk, and I. Richer. A review of the development and performance of the arpanet routing algorithm. *IEEE Trans. on Communications*, COM-26, No. 12:1802–1811, December 1978.

[19] John M. McQuillan, G. Falk, and Ira Richer. A review of the development and performance of the arpanet routing algorithm. *IEEE Trans. on Commun.*, 26(12), December 1978.

[20] Yishay Mansour and Boaz Patt-Shamir. Commpetitive packet routing on grids. In *Proc. of the* 10^{th} *Annual ACM Symposium on Principles of Distributed Computing*, Montreal, Canada (1991).

[21] J.M. McQuillan, I. Richer, and E.C. Rosen. The new routing algorithm for the arpanet. *IEEE Trans. on Communications*, COM-28(5):711–719, May 1980.

[22] S. Plotkin, D. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *Proc. 32nd IEEE Symp. on Found. of Comp. Science*, 1991.

[23] P. Raghavan and C.D. Thompson. Provably good routing in graphs: Regular arrays. In *Proc. 17th ACM Symp. on Theory of Computing*, May 1985.

[24] M. Schwartz and T. E. Stern. Routing techniques used in computer communication networks. *IEEE Trans. on Commun.*, COM-28(4):539–552, April 1980.

[25] E. Upfal. An $o(log n)$ deterministic packet routing scheme. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 241–250, may 1989.