Throughput-Competitive On-Line Routing

Baruch Awerbuch* Lab. for Computer Science MIT

Yossi Azar† Tel-Aviv University and DEC SRC

Serge Plotkin‡ Dept. of Computer Science Stanford University

Abstract

We develop a framework that allows us to address the issues of admission control and routing in high-speed networks under the restriction that once a call is admitted and routed, it has to proceed to completion and no reroutings are allowed. The "no rerouting" restriction appears in all the proposals for future high-speed networks and stems from current hardware limitations, in particular the fact that the bandwidth-delay product of the newly developed optical communication links far exceeds the buffer capacity of the network.

In case the goal is to *maximize the throughput*, our framework yields an on-line $O(\log nT)$-competitive strategy, where n is the number of nodes in the network and T is the maximum call duration. In other words, our strategy results in throughput that is within $O(\log nT)$ factor of the highest possible throughput achievable by an omniscient algorithm that knows all of the requests in advance. Moreover, we show that no on-line strategy can achieve a better competitive ratio.

Our framework leads to competitive strategies applicable in several more general settings. Extensions include assigning each connection an associated "profit" that represents the importance of this connection, and addressing the issue of call-establishment costs.

1 Introduction

Motivation. High-speed integrated communication networks are going to be the most important communication platform of the future. The technological advances in this area are quickly offset by increase in consumption, due to a wide spectrum of new applications (teleconferencing, cable-TV, tele-shopping, etc.). It is thus important to address the fundamental problem of efficient allocation of network resources.

One of the main network resources is the available bandwidth of the communication channels. In order to use the network (say, transmit video signal from one point to another) the user requests a (virtual) connection to be established between these points. Although the rate of information flowing through such a connection might vary in time, the network has to guarantee that the connection will support at least the bit rate that was agreed upon during the connection establishment negotiations. This guarantee is imperative for correct operation of many of the services, including constant bit-rate video and voice transmission. In other words, establishing a connection corresponds to reserving the requested bandwidth along some path connecting the end points specified by the user.

In the context of low-speed networks (*e.g.* IN-

*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.

†DEC System Research Center, 130 Lytton, Palo Alto, CA 94301. Department of Computer Science, Tel-Aviv University, Israel.

‡Department of Computer Science, Stanford University. Research supported by U.S. Army Research Office Grant DAAL-03-91-G-0102, and by a grant from Mitsubishi Electric Laboratories.

TERNET), where buffer requirements are less of an issue, the difficulties associated with on-line circuit-switching may be alleviated by delaying the transmission, slowing down its rate, or by rerouting the connection after it has been established. However, these approaches are usually inappropriate in the context of Gigabit rate networks (*e.g.* ATM [2], PARIS/plaNET [4, 8]). This is mostly due to the fact that the product of transmission rates and network latency well exceeds the available nodal buffer space, in other words because of the high bandwidth-delay product.

In this paper we describe an on-line framework that allows us to address both the admission control (*i.e.* which requests to satisfy and which ones to reject), and bandwidth reservation issues. Our techniques are applicable in the context of real time high-speed networking environments with strict performance guarantees: transmission must start at a specific time, end at a specific time, use a specific amount of bandwidth, and is *guaranteed* to be successfully accomplished, once *admitted* into the network. We assume that requests for establishment of connections arrive on-line; each request specifies the source and destination nodes, the requested bandwidth, and the duration. The algorithm either *rejects* the request, or establishes a connection by allocating the required bandwidth along some path between the source and the destination nodes for the specified duration.

A natural performance measure is the amortized *throughput*, which is defined as the average over time of the number of bits transmitted by the accepted connections.[1] In fact, our framework can be described in terms of a generalization of the throughput. We assume that each request for connection has an associated *profit*, which is received only if the request is satisfied. The goal is to *maximize the profit.* In the simplest case, where the profit is proportional to the bandwidth-duration product, maximizing the total profit corresponds to maximizing the throughput. Roughly speaking, the profit abstraction is useful if the connections differ in importance; in this case one can assign higher profit per bit for the more important connection.

[1]To simplify the definitions, we assume that the customer will use as much bandwidth as he has requested.

Since the admission control and routing algorithm has to make decisions without knowledge of the future requests, it is natural to evaluate its performance in terms of the *competitive ratio* [15], which in this case is the supremum, over all possible input sequences, of the ratio of the profit achieved by the on-line algorithm to the profit achieved by the optimal off-line algorithm. Note that several natural approaches do not lead to algorithms with good competitive ratio. On one hand, a greedy admission strategy that always accepts a connection as long as there exists a path from source to sink with sufficient residual bandwidth may, in some cases, work very poorly since certain admitted connection may end up "blocking" many future, perhaps more profitable, connections. On the other hand, a conservative policy of waiting for most "profitable" connections may clearly lead to very poor performance in cases where only low-profit connections show up.

Our results versus existing work. In this paper we describe an admission and routing strategy that achieves an $O(\log nT)$ competitive ratio if the profit of a call is proportional to the bandwidth-duration product, where T is maximum duration of a call. We also prove that the above bound is tight. We prove similar results for several generalizations of this problem; discussion of these generalizations is deferred to the end of this section.

The techniques used in our algorithm resemble the ones previously used in the context of multicommodity network flow [14, 12], approximate fractional packing [13] and on-line load balancing [1, 6]. In particular, we use of the idea of assigning each edge a cost function that is exponential in its current load, and the idea of concurrently working with multiple copies of the graph, one copy per each time instance.

The first competitive solutions for on-line throughput maximization have been pioneered by Garay and Gopal for the case of a single link [10], and by Garay, Gopal, Kutten, Mansour and Yung [9] for a line network; the latter work achieved logarithmic competitive ratio. The problem we are concerned with, namely competitively maximizing network throughput, has been open for general network topologies. Other previous work on throughput concentrated on *probabilistic* models, and was based on various assumptions on the distributions of call arrivals times and source-destination pairs (see *e.g.* [11]).

Instead of throughput, one can measure "relative load" which is defined as the maximum (over all links and over all moments in time) of the link capacity utilization by the currently routed circuits.[2] Roughly speaking, when we say that the competitive ratio of an on-line algorithm is λ with respect to load means that, given any sequence of requests that can be satisfied by the off-line algorithm without overflowing the capacities, we can satisfy all of these requests on-line if we reduce the rate of each request by a factor of λ. Alternatively, the on-line algorithm can satisfy all of these requests if we increase the capacity of each edge by a factor of λ.

The problem of minimizing the relative load in the context of machine scheduling was considered in [7, 5]. On-line algorithms that are $O(\log n)$ competitive with respect to load for the case of *permanent* virtual circuit routing were presented in [1]. Extension of these techniques to the case of virtual circuits with *known duration* appeared in [6]. Recent results in [3] address the case where the duration of each virtual circuit is a priori unknown. They show that in this case, in order to be competitive, one has to allow to reroute circuits, and present an algorithm that reroutes each circuit at most $O(\log n)$ times while maintaining a competitive ratio of $O(\log n)$ with respect to load.

Generalizations and extensions. The framework presented in this paper leads to on-line algorithms with polylogarithmic competitive ratio in several more general settings. In particular, we show that our algorithm can easily handle the situation where the required bandwidth does not remain constant for the duration of the call, as long as the required bandwidth vs. time function is known during the call negotiation phase. Similarly, the presented algorithm can handle the case where the ratio of profit to the bandwidth-duration product varies from call to call.

An interesting generalization of the model is to incorporate the notion of *call-establishment costs*. Here, the profit accrued by the system is computed as the difference between the profit associated with the request, and a cost which might depend both on the request and on the path used to route the connection. A straightforward modification of the presented algorithm leads to logarithmic competitive ratio in this model. It is also possible to adapt the algorithm to allow negotiation between the network and the customers on the amount of the requested bandwidth. In that case the profit depends, of course, on the amount of bandwidth that the algorithm agrees to reserve for the customer.

An important issue that arises in the context of this work is the question of how to define competitiveness in a very long (or even non-terminating) execution. The disadvantage of the competitive factor as described above is that it provides an amortized measure of performance where the amortization is over all of the time interval in which the system was active. Roughly speaking, by measuring the total profit accrued since time zero, we allow on-line algorithm to grossly misbehave *locally* in certain epochs of the execution's

[2] Note that using relative load as a performance measure makes sense only if we implicitly assume that the off-line algorithm does not need to reject any requests and disallow rejections by the on-line algorithm as well.

history. Intuitively, this is unsatisfactory in the considered context, since the fact that the routing algorithm accrued a lot of profit "last year" should not allow it to reject all the connections during the "next year", if the duration of the requests is measured in days. A natural approach is to compare the performance of the on-line vs. off-line algorithm on any sufficiently long (with respect to the maximum call duration) interval of time, not necessarily starting at time zero.

As it turns out, our algorithm can be proved competitive with respect to this modified measure as well. For example, one of the properties of the presented algorithm is that the profit accrued by the off-line algorithm during any given interval $[\tau_1, \tau_2]$, is within a logarithmic factor of the profit accrued by the on-line algorithm in a slightly larger interval $[\tau_1 - T, \tau_2 + T]$, where T is the maximum duration of a connection. A natural question is why are we not comparing on-line vs. off-line on the same given interval. To address this question, in the full paper we show that for any on-line algorithm and for any time interval, one can always find a sequence of requests where the off-line to on-line profit ratio is unbounded.

2 Preliminaries and Definitions

The network is represented by a capacitated (directed or undirected) graph $G(V, E, u)$. The capacity $u(e)$ assigned to each edge $e \in E$ represents the bandwidth available on this edge. The input sequence consists of a collection of *connection requests*: $\beta_1, \beta_2, \ldots, \beta_k$, where the ith request is represented by the tuple:

$$\beta_i = \left(s_i, t_i, r_i(\tau), T^s(i), T^f(i), \rho(i)\right).$$

Node s_i is the origin of the connection β_i, node t_i is its destination, $r_i(\tau)$ is the function that defines the traffic rate at time τ required by the connection, and $\rho(i)$ is the "revenue" that the algorithm receives if it commits to routing this connection. $T^s(i)$ and $T^f(i)$ are the starting time and completion time, respectively, for the connection. For simplicity, we assume that these times are integer. Upon receiving a connection request β_i, the algorithm either *routes* it by assigning it a path P_i from s_i to t_i, or *rejects* it. In the later case, we set $P_i = \emptyset$. To simplify notation, we assume that $r_i(\tau)$ is defined for any τ, but that $r_i(\tau) = 0$ for $\tau \notin [T^s(i), T^f(i))$. The *relative load* on edge e just before considering the kth request is defined by:

$$\lambda_e(\tau, k) = \sum_{e \in P_i, i<k} \frac{r_i(\tau)}{u(e)}.$$

We require that the capacity constraints will be enforced, *i.e.* $\forall \tau,\ e \in E,\ k: \quad \lambda_e(\tau, k) \leq 1$. The goal of the algorithm is to route *maximum* number of connections weighted by their profits, *i.e.* maximize $\sum_{P_i \neq \emptyset} \rho(i)$.

Another constraint on the algorithm is that it must be *on-line*, in the sense that the decision about routing or rejecting a connection β_i is made at its start point $T^s(i)$, without any knowledge about future connections. Once a connection is made, it cannot be interrupted nor can it be rerouted.

Let $T(j) = T^f(j) - T^s(j)$ be the duration of connection j, and $T = \max_j\{T(j)\}$ be the maximum duration of a connection. As mentioned in the introduction, we consider the case where the profit of each request is proportional to the rate and to the duration of this request. In fact, we allow some variation in the profit for a unit of rate for a unit of time, as long as this variation is not very large. More precisely, we normalize the profit such that for any connection β_j and $r_j(\tau) \neq 0$ we have:

$$(1) \qquad 1 \leq \frac{1}{n} \cdot \frac{\rho(j)}{r_j(\tau)T(j)} \leq F.$$

Note that $1/n$ factor in above inequalities is used only for convenience of normalization. One interesting special case is when the requested rate is constant per connection, and when the profit is exactly proportional to the rate-duration product, *i.e.* to the number of bits that can be sent using this connection. For this case, we have $F = 1$.

Denote $\mu = 2nTF+1$. We assume that for any j and τ,

$$(2) \qquad r_j(\tau) \le \frac{\min_e\{u(e)\}}{\log \mu}.$$

Informally, this means that the requested rates are significantly smaller than the minimum available capacity in the network. Although, at first glance, this restriction seems somewhat artificial, in Section 4 we show that without this restriction it is impossible to design on-line algorithms with polylogarithmic competitive ratio.

3 The Admission Control and Routing Algorithm

The admission control and routing algorithm ROUTE_OR_BLOCK is shown in Figure 1. Consider $T^s(j)$, the start time of request β_j. With each edge e and time instance τ, we associate a "cost" of this edge, defined by $c_e(\tau, j) = u(e)(\mu^{\lambda_e(\tau,j)} - 1)$. The algorithm routes β_j on a path that is small with respect to a weighted average of these costs. More precisely, if $e \in P_j$, then e's contribution to the cost of the path is computed as: $\sum_\tau \frac{r_j(\tau)}{u(e)} c_e(\tau, j)$. If there exists a path which cost is bounded by the profit $\rho(j)$, then this path is used to route the connection β_j. Otherwise, the connection is rejected.

The analysis of the algorithm is divided into two parts. First, we prove that the algorithm does not violate the capacity constraints, and then we show that the profit accrued by the algorithm is within a logarithmic factor of the profit accrued by the optimal off-line algorithm.

Informally, the reason that the capacity constraints are always satisfied, is that when an edge

NEW_CONNECTION$(s, t, T^s, T^f, r(\tau), \rho)$:

$\forall \tau, e \in E: \; c_e(\tau, j) \leftarrow u(e)(\mu^{\lambda_e(\tau,j)} - 1)$;

if $\exists$ path P in $G(V, E)$ from s to t s.t. $\sum_\tau \frac{r(\tau)}{u(e)} c_e(\tau, j) \le \rho$
then <u>route</u> the connection on P, and set:
$\forall e \in P, T^s \le \tau \le T^f$,
$\lambda_e(\tau, j+1) \leftarrow \lambda_e(\tau, j) + \frac{r(\tau)}{u(e)}$
else <u>block</u> the connection

Figure 1: The ROUTE_OR_BLOCK Algorithm.

that is close to being saturated, its cost is high enough that it will never be used for routing. Let $\mathcal{A}$ denote the set of indices of requests that were satisfied by ROUTE_OR_BLOCK, *i.e.* $\mathcal{A} = \{i : P_i \neq \emptyset\}$.

Lemma 3.1 For all edges $e \in E$ and all times τ, $\sum_{i \in \mathcal{A}, e \in P_i} r_i(\tau) \le u(e)$.

Proof: Let β_j be the first connection that was assigned to an edge e and that caused relative load to exceed 1. In other words, e has available capacity less than $r_j(\tau)$ at some instance τ where $T^s(j) \le \tau \le T^f(j)$. By the definition of relative load, we have $\lambda_e(\tau, j) > 1 - \frac{r_j(\tau)}{u(e)}$. Using the assumption that $r_j(\tau) \le \frac{u(e)}{\log \mu}$, we get:

$$\begin{aligned} c_e(\tau, j)/u(e) &= \mu^{\lambda_e(\tau,j)} - 1 \\ &\ge \mu^{1 - \frac{1}{\log \mu}} - 1 \\ &= \mu/2 - 1 = TFn \end{aligned}$$

Therefore, using Assumption (1):

$$\frac{r_j(\tau)}{u(e)} c_e(\tau, j) \ge TFn \cdot r_j(\tau) \ge \rho(j).$$

Hence, connection j could not have used link e. ∎

The next lemma shows that we can use sum of link costs to lower-bound the total profit accrued by our algorithm.

Lemma 3.2 Let $\mathcal{A}$ be the set of indices of connections routed by ROUTE_OR_BLOCK algorithm, and let k be the index of the last connection. Then

$$2\log\mu \sum_{j\in\mathcal{A}} \rho(j) \geq \sum_{\tau}\sum_{e} c_e(\tau, k+1).$$

Proof: By induction on k. For $k = 0$ the inequality is trivially true since both sides are 0. Connections that were refused do not change either side of the inequality. Thus, it is enough to show that, for any j, if we admit connection β_j, we get:

$$\sum_{\tau}\sum_{e}[c_e(\tau, j+1) - c_e(\tau, j)] \leq 2\rho(j)\log\mu.$$

Consider link $e \in P_j$. Using the definition of the link cost, we get:

$$\begin{aligned} c_e(\tau, j+1) - c_e(\tau, j) &= \\ & u(e)(\mu^{\lambda_e(\tau,j)+\frac{r_j(\tau)}{u(e)}} - \mu^{\lambda_e(\tau,j)}) \\ &= u(e)\left(\mu^{\lambda_e(\tau,j)}(\mu^{\frac{r_j(\tau)}{u(e)}} - 1)\right) \\ &= u(e)\left(\mu^{\lambda_e(\tau,j)}(2^{\log\mu\frac{r_j(\tau)}{u(e)}} - 1)\right) \end{aligned}$$

By Assumption (2), we have $r_j(\tau) \leq \frac{u(e)}{\log\mu}$. Since $2^x - 1 \leq x$ for $0 \leq x \leq 1$, we conclude

$$\begin{aligned} c_e(\tau, j+1) - c_e(\tau, j) &\leq \mu^{\lambda_e(\tau,j)} r_j(\tau)\log\mu \\ &= \left(c_e(\tau, j)\frac{r_j(\tau)}{u(e)} + r_j(\tau)\right)\log\mu. \end{aligned}$$

The above upper bound on the change in costs, the fact that the connection β_j was admitted, and Assumption (1) imply:

$$\begin{aligned} &\sum_{\tau}\sum_{e}[c_e(\tau, j+1) - c_e(\tau, j)] \\ &\quad\leq \log\mu \sum_{\tau}\sum_{e\in P_j}\left(c_e(\tau, j)\frac{r_j(\tau)}{u(e)} + r_j(\tau)\right) \\ &\quad\leq \log\mu\left(\rho(j) + \sum_{\tau}|P_j|\cdot r_j(\tau)\right) \\ &\quad\leq 2\rho(j)\log\mu. \end{aligned}$$

∎

Next we show that sum of the link costs is an upper bound on the maximum profit that can be obtained by the optimal off-line algorithm.

Lemma 3.3 Let $\mathcal{Q}$ be the set of indices of the connections that were admitted by the off-line algorithm but not by the on-line algorithm, and denote $\ell = \max\{\mathcal{Q}\}$. Then $\sum_{j\in\mathcal{Q}}\rho(j) \leq \sum_{\tau}\sum_{e} c_e(\tau, \ell)$.

Proof: Let P'_j be the path used by the off-line algorithm to route β_j, for $j \in \mathcal{Q}$. The fact that β_j was not admitted and monotonicity in j of the costs $c_e(\tau, j)$ imply

$$\begin{aligned} \rho(j) &\leq \sum_{\tau}\sum_{e\in P'_j} r_j(\tau)c_e(\tau, j)/u(e) \\ &\leq \sum_{\tau}\sum_{e\in P'_j} r_j(\tau)c_e(\tau, \ell)/u(e). \end{aligned}$$

Summing over all $j \in \mathcal{Q}$, we get:

$$\begin{aligned} \sum_{j\in\mathcal{Q}}\rho(j) &\leq \sum_{j\in\mathcal{Q}}\sum_{\tau}\sum_{e\in P'_j}\frac{r_j(\tau)}{u(e)}c_e(\tau, \ell) \\ &\leq \sum_{\tau}\sum_{e} c_e(\tau, \ell)\sum_{j\in\mathcal{Q}, e\in P'_j}\frac{r_j(\tau)}{u(e)} \\ &\leq \sum_{\tau}\sum_{e} c_e(\tau, \ell). \end{aligned}$$

The last inequality follows from the fact that the off-line algorithm cannot exceed unit relative load at any instance in time. ∎

Theorem 3.4 The ROUTE_OR_BLOCK Algorithm, shown in Figure 1, never violates the capacity constraints and accrues at least $\frac{1}{2\log(2\mu)}$-fraction of the profit accrued by the optimal off-line algorithm.

Proof: The profit accrued by the off-line algorithm can be bounded from above by:

$$\sum_{i\in\mathcal{Q}}\rho(i)+\sum_{i\in\mathcal{A}}\rho(i)$$

Using Lemma 3.3, this profit is upper-bounded by:

$$\sum_{\tau}\sum_{e}c_e(\tau,\ell)+\sum_{i\in\mathcal{A}}\rho(i).$$

Since $c_e(\tau,k+1)$ is the final cost of the edge e, we know that $\forall e\in E, c_e(\tau,k+1)\geq c_e(\tau,\ell)$. Together with Lemma 3.2, this implies that the profit of the off-line is bounded by

$$\begin{aligned}2\log\mu\sum_{i\in\mathcal{A}}\rho(i)+\sum_{i\in\mathcal{A}}\rho(i)&\\ &\leq(2\log\mu+1)\sum_{i\in\mathcal{A}}\rho(i)\\ &\leq 2\log(2\mu)\sum_{i\in\mathcal{A}}\rho(i).\end{aligned}$$

The above bound on the accrued profit, together with Lemma 3.1, complete the proof of the claim. ∎

Remark: As we have mentioned in the Introduction, it is interesting to compare the off-line and on-line profits performance over an arbitrary interval in time that does not start at time zero. More precisely, let $[\tau_1,\tau_2]$ be some interval in time and consider the profit $\rho^{\text{off}}[\tau_1,\tau_2]$ obtained by the off-line algorithm due to requests $\{\beta_j : T^s(j)\in[\tau_1,\tau_2]\}$, and let $\rho^{\text{on}}[\tau_1,\tau_2]$ be the corresponding profit of the on-line algorithm. In the full paper we show that $\rho^{\text{on}}[\tau_1-T,\tau_2+T]$ is within a logarithmic factor of the $\rho^{\text{off}}[\tau_1,\tau_2]$. Roughly speaking, this implies that the off-line profit on a given interval is not much higher than the corresponding on-line profit on a slightly larger interval. We also show that for any on-line algorithm and for any time interval, one can always find a sequence of requests where the ratio of $\rho^{\text{off}}[\tau_1,\tau_2]/\rho^{\text{on}}[\tau_1,\tau_2]$ is unbounded.

4 The Lower Bounds

In this section we show that our algorithm is optimal with respect to the achieved competitive ratio. We also justify our assumption of bounding the rates of the requests. First we show that even if all the requested rates are very small, the profit accrued by the off-line algorithm can exceed the best possible on-line profit by at least an $\Omega(\log\mu)$ factor. In all of the subsequent proof we assume that all the capacities are 1, and that all requests appear at the beginning. Moreover, we assume that all the requests have some fixed rate α. In the end of this section, we show much stronger bounds for the case where α is allowed to be large relative to the capacities in the network.

Let $G(n)$ be a graph which is a line of n edges ($n+1$ vertices). Denote the vertices by $v_0,\ldots,v_n$, and let n be a power of 2.

Lemma 4.1 Any on-line algorithm for $G(n)$ has competitive ratio of $\Omega(\log n)$.

Proof: Let all requests have unit duration. Consider sequence of requests that consists of $\log n+1$ phases. Each phase i, for $0\leq i\leq\log n$, consists of 2^i groups of requests, $0\leq j\leq 2^i-1$. A request in phase i, group j has $v_{jn/2^i}$ as its starting node and $v_{(j+1)n/2^i}$ as its destination. For each i,j there are $1/\alpha$ identical requests, each requesting capacity α and providing the same profit, say α.

Let x_i be the amount of profit that the on-line algorithm accrues due to the requests in phase i. A unit of profit due to requests in phase i can be achieved only by using up $n/2^i$ units of capacity. Since there are only n units of capacity overall, we get: $\sum_{i=0}^{\log n} 2^{-i} n x_i \leq n$. Define $S_j = 2^{-j} \sum_{i=0}^{j} x_i$. Then,

$$\sum_{j=0}^{\log n} S_j = \sum_{0 \leq i \leq j \leq \log n} 2^{-j} x_i \leq \sum_{i=0}^{\log n} 2 \cdot 2^{-i} x_i \leq 2$$

Hence, there exists k such that $S_k \leq 2/\log n$. Now consider a prefix consisting of the first k phases of the request sequence. The benefit of the on-line algorithm in this case is $\sum_{i=0}^{k} x_i = 2^k S_k \leq 2^k \cdot (2/\log n)$. The off-line algorithm can reject all the requests except the ones in phase k, accruing benefit of 2^k. ∎

Consider a case where the graph consists of a single link. By constructing a sequence of requests that have exponentially growing duration from phase to phase and another sequence of requests where the profit per transmitted bit increases exponentially with the phase number, it is relatively easy to show the following claim:

Lemma 4.2 Any on-line algorithm has competitive ratio of $\Omega(\log(TF))$.

By combining the Lemmas 4.2 and 4.1, we get the following theorem.

Theorem 4.3 Any on-line algorithm has throughput competitive ratio of $\Omega(\log nFT)$

Next we show that if some connections request rates in excess of $1/k$-factor of the capacity, then the competitive ratio of any on-line algorithm is bounded by $\Omega(T^{1/k} + F^{1/k} + n^{1/k})$. In other words, in order to achieve polylogarithmic competitive ratio, we need to bound the maximal rate of a connection to be below $\log(TFn)/\log\log(TFn)$ fraction of the minimum capacity. This bound is close to the $\log(TFn)$-fraction bound that, as shown in Section 3, implies logarithmic competitive ratio of the ROUTE_OR_BLOCK algorithm.

Lemma 4.4 If we allow requests of rate as large as $1/k$-fraction of the minimum capacity, then the competitive ratio is at least $\Omega(T^{1/k} + F^{1/k})$ for any algorithm.

Proof: Omitted.

For the next lower bound we use $G(n)$ which is a line of n unit-capacity edges, where n be a power of 2. All requests have unit duration.

Lemma 4.5 If we allow requests of rate $1/k$ then the competitive ratio is $\Omega(n^{1/k})$ for any algorithm for $G(n)$.

Proof: The sequence of requests consists of $k+1$ phases. In phase 0 there is a request from v_0 to v_n. The first request must be accepted since it might be the only request. Thus before phase 1 the utilization of each edge is $1/k$ and the benefit of the on-line is $1/k$. For phases $1 \leq i \leq k$ we maintain the following invariant. Either the lower bound was already proved or before phase i the utilization of each edge in the range l_i to $l_i + n^{1-(i-1)/k}$ is i/k and the total benefit of the on-line for all previous requests is only i/k.

Let $l_1 = 0$. The invariant clearly holds for $i = 1$. We assume by induction that the invariant is true for i and some l_i and we prove it for $i+1$ and some l_{i+1}. We make k requests between $v_{l_i + jn^{1-i/k}}$ to $v_{l_i+(j+1)n^{1-i/k}}$ for each $0 \leq j < n^{1/k}$. If the on-line algorithm reject all of these request we are done, since the off-line algorithm can accept all the request in the phase and get benefit $n^{1/k}$. The benefit of the on-line algorithm in this case is bounded by $i/k \leq 1$ which implies the lower bound.

On the other hand if some request was accepted, then we stop phase i after that first such request. That request is between $v_{l_i+jn^{1-i/k}}$ to $v_{l_i+(j+1)n^{1-i/k}}$. Define $l_{i+1} = l_i + jn^{1-i/k}$. We claim that the invariant holds for $i+1$. The total benefit of the on-line algorithm before phase $i+1$ is $i/k + 1/k = (i+1)/k$; the utilization of the edges from l_{i+1} to $l_{i+1} + n^{1-i/k}$ is $i/k + 1/k = (i+1)/k$.

This completes the proof of the invariant.

The invariant implies that at the start of phase k, all the edges from l_k to $l_k + n^{1/k}$ are fully utilized. Thus, the the on-line algorithm has to reject all the requests from v_{l_k+j} to v_{l_k+j+1} $0 \leq j < n^{1/k}$. In contrast to this, the off-line algorithm accepts all of these requests, and gets benefit $n^{1/k}$. The claim follows since the on-line benefit is bounded by 1. ∎

By combining the Lemmas 4.4 and 4.5, we get the following theorem:

Theorem 4.6 If the rate of requests can be as large as $1/k$ of the capacities then throughput competitiveness of any on-line algorithm is $\Omega(T^{1/k} + F^{1/k} + n^{1/k})$.

Acknowledgements

We would like to thank Orli Waarts for many helpful discussions.

References

[1] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line machine scheduling with applications to load balancing and virtual circuit routing. In *Proc. 25th Annual ACM Symposium on Theory of Computing*, pages 623–631, May 1993.

[2] Special issue on Asynchronous Transfer Mode. *Int. Journal of Digital and Analog Cabled Systems*, 1(4), 1988.

[3] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Competitive routing of virtual circuits with unknown duration. Unpublished manuscript, July 1993.

[4] B. Awerbuch, I. Cidon, I. Gopal, M. Kaplan, and S. Kutten. Distributed control for PARIS. In *Proc. 9th Annual ACM Symposium on Principles of Distributed Computing*, pages 145–160, 1990.

[5] Y. Azar, A. Broder, and A. Karlin. On-line load balancing. In *Proc. 33rd IEEE Annual Symposium on Foundations of Computer Science*, pages 218–225, 1992.

[6] Y. Azar, B. Kalyanasundaram, S. Plotkin, K. Pruhs, and O. Waarts. On-line load balancing of temporary tasks. In *Proc. Workshop on Algorithms and Data Structures*, August 1993.

[7] Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignment. In *Proc. 3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 203–210, 1992.

[8] I. Cidon and I. S. Gopal. PARIS: An approach to integrated high-speed private networks. *International Journal of Digital & Analog Cabled Systems*, 1(2):77–86, April-June 1988.

[9] J. Garay, I. Gopal, S. Kutten, Y. Mansour, and M. Yung. Efficient on-line call control algorithms. In *Proc. of 2nd Annual Israel Conference on Theory of Computing and Systems*, 1993.

[10] J.A. Garay and I.S. Gopal. Call preemption in communication networks. In *Proc. INFOCOM '92*, volume 44, pages 1043–1050, Florence, Italy, 1992.

[11] F. P. Kelly. Blocking probabilities in large circuit-switched networks. *Advances in Appl. Prob.*, 18:473–505, 1986.

[12] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problem. In *Proc. 23th ACM Symposium on the Theory of Computing*, pages 101–111, May 1991.

[13] S. Plotkin, D. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *Proc. 32nd IEEE Annual Symposium on Foundations of Computer Science*, pages 495–504, October 1991.

[14] F. Shahrokhi and D. Matula. The maximum concurrent flow problem. *J. Assoc. Comput. Mach.*, 37:318–334, 1990.

[15] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. ACM*, 28(2):202–208, 1985.