Efficient On-Line Call Control Algorithms

Juan A. Garay [*] Inder S. Gopal [†] Shay Kutten [†]
Yishay Mansour [†] Moti Yung [†]

Abstract

We study the problem of on-line *call control* in a communications network, i.e., the problem of accepting or rejecting an incoming call (a request for a connection between two points in a network) without having the knowledge of future calls. The problem is a part of the more general problem of bandwidth allocation and management. Intuition suggests that knowledge of future call arrivals can be crucial to the performance of the system. In this paper, however, we present preemptive deterministic on-line call control algorithms. We use competitive analysis to measure their performance—i.e., we compare our algorithms to their off-line, clairvoyant counterparts—and prove optimality for some of them.

In this paper we consider two specific networks: a line of nodes and a single edge, and investigate a variety of cases concerning the *value* of the calls. The value is accrued only if the call terminates successfully; otherwise—if the call is rejected, or prematurely terminated—no value is gained. The performance of the algorithm is then measured by the cumulative value achieved, when given a sequence of calls. The variety of call value criteria that we study—constant; proportional to the length of the call's route; proportional to its holding time—captures many of the natural cost assignments to network services.

[*]IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598.
[†]Computer Science Department, Tel-Aviv University, Israel.

1 Introduction

High-speed networks (e.g., [14, 5, 9, 8]), as providers of multimedia services, will have to support a wide variety of traffic types. Each of these traffic types will have very different requirements in terms of required duration and throughput, and tolerable delay and message loss. Many of these requirements will only be met if bandwidth can be guaranteed for the individual connections in accordance with their requirement.

Thus, bandwidth reservation and management is a central issue in network control and operation. Roughly speaking, its objective is to maximize the usage of the network facilities, while minimizing the probability that a particular connection will be denied access to the network, or "blocked". This issue is one of the most actively studied open problems in the area of high-speed networks. The nature of the problem is such that it is always possible to "second guess" decisions made in the past. In other words, a decision made previously to accept a connection[1] may have been wrong because it caused a subsequent more "valuable" call to be rejected. Thus, the *on-line* nature of the problem, namely the fact that decisions are to be made when calls arrive into the system without knowledge of future arrivals, might lead to significantly lower efficiency than would have been possible with full *off-line* knowledge of the entire pattern of call arrivals.

This aspect of the problem leads naturally to the investigation of the issue of call *preemption* from an on-line perspective. In other words, if previous decisions were incorrect, it may pay to attempt to rectify them by preempting or removing a call in progress from the network. For example, if a high capacity call is blocked because the capacity used up by a low capacity call, it may be worthwhile to preempt the low capacity call and accept the high capacity call even at the risk of interrupting a call in progress.

In this paper we investigate this issue, i.e., we study call preemption algorithms in an on-line fashion. Generally speaking, an on-line problem is one in which an algorithm that solves it must handle a sequence of requests, handling each request without knowledge of the future requests. Examples of on-line problems include the scheduling of jobs on (parallel) machines, paging—i.e., the allocation of cache memory, the maintenance of dynamic data structures, network routing, etc.

We evaluate on-line algorithms in terms of their *competitiveness* (a measure introduced in [13], and thereafter intensively applied to various on-line settings). An algorithm is said to be c-competitive, $0 < c \leq 1$, if its performance on *any* sequence of requests is within a factor c of the performance of any other algorithm on the same sequence, including the off-line, clairvoyant algorithms for the problem, which can "see" all future requests. A main virtue of an efficient competitive algorithm is its robustness, i.e., the fact that the algorithm works well for any distribution of the requests, and no assumptions need to be made about them. Another attractive aspect of on-line algorithms is their simplicity, since typically they do not involve heavy processing of past history. The bounds are sometimes best explained as a game between a player (the on-line algorithm) and an

[1] We use the terms *call* or *connection* interchangeably.

adversary, who generates part of the request sequence, observes the player's response to it, and then extends the sequence by producing more requests.

An initial study of the call control problem in an on-line fashion was presented by Garay and Gopal in [11]. One of the cases studied there was the maximization of line utilization on a single edge for a call value given by the call's *holding time*.[2] They showed that if the holding times of the arriving calls are unknown, then competitive algorithms (deterministic or randomized) do not exist. Roughly speaking, this is so because the adversary can always orchestrate situations in which the algorithm, unaware of the calls' duration, ends up preempting (or rejecting) long-duration calls in favor of short ones, thus yielding an unbounded competitive ratio. It was also shown in [11] that if there is *no penalty* for preempting an existing call (e.g., the telephone company charges the customer according to the usage time, even if the call is disconnected), then an optimal 1-competitive algorithm exists. The case when there is a penalty for preempting a call (or more precisely, no value is accrued when a call is preempted) was left open.

This is one of the problems we address in this paper. Namely, we show that when the penalty for preemption is the call's holding time, then an optimal competitive algorithm for a single edge exists. (This case makes sense in practice, as losing utilization implies the loss of service revenues.) In other words, our online algorithm can only gain if a call is completed, and does not accrue anything for a partial call. We also investigate two other cases of value assignment to calls for a network consisting of a line of nodes (constant, and proportional to the length of the call's route), and provide competitive algorithms for theses cases as well. We believe that the variety of call values we cover captures the most natural cost assignments to network services.

In order to highlight the issues involved in the call control problem, our models simply consist of a single edge (a direct communication link, or, more generally, a communication subsystem) interconnecting two nodes (representing processors, transmission stations, gateways, etc.), and of a collection of nodes interconnected by a communication line. The term "call" refers to any communication application requiring a connection (and bandwidth) in an interval between two nodes. Clearly, our abstraction does not comprise a complete investigation of this on-line problem for general networks; we view our contribution as a step towards a better analytical understanding of call management in communication networks.

Indeed, our work has, directly or indirectly, motivated a number of recent interesting works. One immediate direction is investigating the problem in other network topologies. This has been addressed by Awerbuch, Bartal, Fiat and Rosén in [3], where they give a competitive (randomized) algorithm for non-preemptive call scheduling on tree networks. Awerbuch, Gawlick, Leighton and Rabani [4] have considered also non-tree networks, such as meshes. Awerbuch, Azar and Plotkin [1] provide a competitive call control strategy for general networks if the profit of a call is proportional to the bandwidth-duration product.

[2]The holding time of a call is the time that elapses from its creation until it terminates. Although it is not always possible in reality for the algorithm to "know" this parameter at the time a call arrives, in many circumstances this information can be available, or estimated with overwhelming probability, from, for example, statistical history traces, the call type (e.g., voice, video), etc. In this paper we assume that the available estimates are correct.

Awerbuch, Azar, Plotkin and Waarts [2] study the case of unknown call duration, and show how to achieve competitiveness by allowing rerouting of the calls. More recently, Bar-Noy *et al.* [6] have considered the problem where the bandwidth requirement is general, but impose the restriction that the maximum bandwidth required by a single call is at most a constant fraction (say, half) of the link capacity. For this model they present constant-competitive algorithms. See [12] for a recent overview of competitive call admission and routing algorithms.

1.1 Our Results

A communication network is a general graph, $G = (V, E)$, $|V| = n$. In this paper we assume that the routing for each call is defined by a process that is outside of the scope of the call control algorithm. We assume also that E is partitioned into a set of paths, or "lines." The fixed tour taken by any communication application can then be viewed as a collection of segments, each residing entirely on some line. We call a segment a "call" or a "connection," and assume that the management of each line is performed autonomously. Thus, in each line calls "arrive" with some predefined bandwidth and a predefined route—an interval within the line; this assumption gives a simple enough environment to start investigating the involved issues of call control. We remark that the statements we will be making about our algorithms' performance (competitive ratio) will only apply to these lines as, since our algorithms will be scheduling calls autonomously on each line, it is possible that the performance of the overall network could be poor even though each line is doing well.

We use the notation c_i to indicate the ith call arrival. Each call c_i is determined by the tuple $< a_i, d_i, r_i, b_i >$, where a_i and d_i are the call's arrival and holding times, respectively, b_i is the bandwidth, and r_i is the route (segment). The route r_i is an interval, i.e., a set of consecutive edges $\{e_{i_1}, e_{i_2}, \ldots\}$ from the line. We also assume bidirectional edges and capacity assignments. In general, each edge, $e \in E$, has a capacity $C(e)$ associated with it. If a call is accepted into the system, it must have the property that for each edge used by the call the sum of the bandwidth of all the calls that share the edge must be less than the capacity of the edge. (This includes the bandwidth of the newly added call.) Otherwise, the call must be rejected or some of the preexisting calls preempted. In this paper we abstract out the conflict situation by assuming that the capacity of an edge cannot accommodate more than one call at a time, i.e., $C(e) < b_i + b_j$, for all edges e and calls c_i, c_j. For this reason we may assume that $C(e) = 1$ for all edges and $b_i = 1$ for all calls.

For each call c_i, we use $\mathcal{P}_{c_i}$ to denote the set of calls that would have to be preempted in order to accommodate c_i. Given the assumption above, $\mathcal{P}_{c_i}$ consists of the calls whose routes intersect c_i's. It can also be the case that the call control algorithm decides not to accept an incoming call. Specifically, we say a call c_i is *rejected* due to call c_j if when c_i is issued, call c_j is being served, and c_j is the reason that c is not accepted. With each call c_i we associate a function $\mathtt{Val}(c_i)$, which yields a positive value if the call is allowed to terminate, otherwise 0 (for example, telephone companies typically reimburse their customers for disconnected calls; this is our *penalty* for preempting a call). The performance of the algorithm is then measured by the value cumulatively achieved, when given any (finite) sequence of calls.

In this paper we consider the following forms of `Val`, and obtain the following corresponding results:

- $\texttt{Val}(c_i) = |r_i|$. I.e., the value is given by the number of edges of the call (in our setting, this is equivalent to the total bandwidth requested by the call). The network under consideration is a line of nodes, and we assume that both the holding time and the arrival time for all the calls are the same.[3] We give an algorithm that is $\frac{1}{2g+1} \approx 0.24$-competitive, where g is the *golden ratio* (i.e., $g = \frac{1+\sqrt{5}}{2} \approx 1.62$). This bound is optimal, as recently shown by Furst and Tomkins [10].

- $\texttt{Val}(c_i) = O(1)$. The value of a call is constant, and the network consists of a line of processors. As in the previous case, we assume that both the holding time and the arrival time for all the calls are the same. We give an algorithm that is $O(\frac{1}{\log n})$-competitive, and show optimality by providing a matching lower bound that holds for any on-line algorithm.

- $\texttt{Val}(c_i) = d_i$. I.e., the value to be achieved by the completion of the call is (proportional to) the call's holding time. We assume in this case that an estimate of the holding times of each call is available at the call's arrival time. The abstract scenario we consider here is that of two nodes connected by a single communication line. We give an algorithm that is $\frac{1}{4}$-competitive. Optimality follows from reducing our problem to a special machine scheduling problem, and applying the results of [7].

The general form of our algorithms for the three cases above is depicted in Figure 1, where α and β are constants and f and h are functions. The rest of the paper is dedicated to the different value criteria and to the corresponding choice of parameters α, β, f and h that would yield the competitive factors outlined above. Note that since we assume that no two calls can share a link, the set $\mathcal{P}_{c_{new}}$ is unique, i.e. $\mathcal{P}_{c_{new}}$ includes all the calls which share an edge with c_{new} and were previously accepted.

We remark the simplicity of our control structure. This attractive property is crucial in high-speed network environments, where fast services are typically required.

2 Call Value is Length of Route

The model for this section is a network consisting of a line of nodes, and the value of a call is given by the number of links that it occupies, i.e., $\texttt{Val}(c_k) = |r_k|$. This model

[3] This models the case where the call arrivals happen to be close, or where all the calls overlap in time. In fact, this model is equivalent to what other authors have called the "infinite-duration" model, in which arrival times may vary, but holding times are infinite.

{ Let c_{new} be the incoming call. }

if $\alpha \cdot f(c_{new}) > \beta \cdot h(\mathcal{P}_{c_{new}})$ **then**

Terminate($\mathcal{P}_{c_{new}}$);
Accept(c_{new})

else

Reject(c_{new}).

Figure 1: The general form of the call control algorithms.

captures situations in which the server provider charges proportionally to the distance of a connection, or to the total bandwidth allocated. We also assume that calls occupy consecutive links. In this model, we will sometimes refer to r_k as c_k's "interval" when it facilitates the reasoning. We will also refer to the number of links separating the end points of the two calls as the "distance" between the calls.

As mentioned before, we assume that no two calls can occupy the same communication link. Naturally, we also assume that the size of the network is bounded (it can be easily shown that competitiveness is not possible otherwise); alternatively, we can assume for the sake of analysis that there is an upper bound on $\texttt{Val}(c_k)$, for every c_k. We assume that all the calls have the same duration, one time unit, and that they all arrive at the same time, but still need to be handled in an on-line fashion, that is, the call control algorithm has to make the decision for a given call before the parameters of the next call are known.

We call this section's algorithm `LR` (for *length* of *route*); basically, the algorithm accepts an incoming call only if the length of the call is greater than g times the length of any call that must be pre-empted to schedule the incoming call, where g is the *golden ratio* (i.e., $g = \frac{1+\sqrt{5}}{2}$). That is, for the value criterion of this section, we set in the general framework of Figure 1, we set $\alpha = 1$, $\beta = g$, $f = \texttt{Val}$, and $h = \max_{c_i \in \mathcal{P}_{c_k}} \texttt{Val}(c_i) = \max_{c_i = <0,1,r_i,1> \in \mathcal{P}_{c_k}} |r_i|$. The following theorem states the competitiveness of `LR` .

Theorem 2.1 *Algorithm* `LR` *is* $\frac{1}{2g+1}$*-competitive.*

Proof: For each n, $n \geq 1$, let $ACTIVE(n)$ be the set of calls which `LR` has running after the nth call is introduced. We know that this set is never empty, since when the nth call is scheduled, it is either accepted or is rejected due to some other call which is running.

For each call $c_k \in ACTIVE(n)$, we denote as $\mathcal{P}^*_{c_k}$ the transitive closure of $\mathcal{P}_{c_k}$, that is, the set of calls that c_k preempted, the set of calls preempted by these calls, and so forth. (Note that $\mathcal{P}^*_{c_k}$ is defined at the time that call c_k is introduced and does not change with the arrival of future calls.) In addition, for each call $c_k \in ACTIVE(n)$ we denote by $\mathcal{R}_{c_k}(n)$ the set of calls, up to the nth call, that where rejected due to calls in $\mathcal{P}^*_{c_k} \cup \{c_k\}$. (A call c_n is rejected due to call c_k if call $c_k \in \mathcal{P}_{c_n}$ has the maximum value in this set, i.e., $\texttt{Val}(c_k) = \max_{c_i \in \mathcal{P}_{c_n}} \texttt{Val}(c_i)$.) Let $\mathcal{I}_{c_k}(n)$ denote the interval (in communication links) that is the union of the routes of all the calls in $\mathcal{P}^*_{c_k}$, $\mathcal{R}_{c_k}(n)$, and of c_k itself. We will show that for every $c_k \in ACTIVE(n)$, the ratio between $|\mathcal{I}_{c_k}(n)|$ and $\texttt{Val}(c_k)$ is at most $1+2\beta$. Since this ratio also holds after the last call, it implies that the competitiveness of LR is at least $\frac{1}{2\beta+1}$.

We prove this claim by induction on the number of calls. The induction hypothesis is that for each call $c_k \in ACTIVE(n)$ whose interval route is between "point" (i.e., node) y and point $y+x$, then $\mathcal{I}_{c_k}(n)$ is an interval and is contained in $[y - x\beta, y + x + x\beta]$. The base of the induction is clear.

Consider the $(n+1)$st call c_{n+1}. There are a few possibilities for the behavior of LR on this call. The first is that the route of c_{n+1} does not intersect the route of any other call, i.e., $\mathcal{P}_{c_{n+1}}$ is empty. In such a case LR will accept c_{n+1} and no call would be preempted. This means that $ACTIVE(n+1) = ACTIVE(n) \cup \{c_{n+1}\}$. By the induction hypothesis, for each call $c_k \in ACTIVE(n)$ the claim holds. For c_{n+1} the inductive hypothesis holds trivially.

The second case is that the call c_{n+1} is rejected by LR due to call c_k (i.e., c_k is the call with the maximum value in $\mathcal{P}_{c_{n+1}}$). We would like to show that in such a case the route of c_{n+1} is contained within $[y - x\beta, y + x + x\beta]$, where call c_k is between y and $y+x$. Assume to the contrary that this is not true. Then either c_{n+1} includes a point to the left of $y - x\beta$, or a point to the right of $y + x + x\beta$. In either case, this implies that the length of the call c_{n+1} is strictly larger than $x\beta$. But in such a case c_k could not cause c_{n+1} to be rejected. This contradicts the assumption that c_k causes c_{n+1} to be rejected. Therefore $\mathcal{I}_{c_k}(n+1)$ will be contained in $[y - x\beta, y + x + x\beta]$, and the induction hypothesis holds.

Finally, the third case is that call c_{n+1} is accepted by LR and the set of calls $\mathcal{P}_{c_{n+1}}$ is preempted. Assume that the route of call c_{n+1} is between points y and $y+x$. In such a case we first would like to prove that for any call $c_k \in \mathcal{P}_{c_{n+1}}$, the interval $\mathcal{I}_{c_k}(n)$ is contained in $[y - x\beta, y + x + x\beta]$. Assume that call c_k is between w and $w+v$. By the induction hypothesis, $\mathcal{I}_{c_k}(n)$ is contained in $[w - v\beta, w + v + v\beta]$. Since c_k intersects with c_{n+1}, either $y < w < y + x$ or $y < w + v < x + y$. Therefore, $[w - v\beta, w + v + v\beta]$ is contained within $[y - v - v\beta, y + x + v + v\beta]$. Since c_k is being preempted by LR , then $v\beta \leq x$. By the choice of β as the golden ratio, we have that $v + v\beta \leq x/\beta + x = x\beta$. This implies that $\mathcal{I}_{c_k}(n)$ is contained in $[y - x\beta, y + x + x\beta]$.

By definition $\mathcal{I}_{c_{n+1}}(n+1)$ denote the interval (in communication links) that is the union of the routes of all the calls in $\mathcal{P}^*_{c_{n+1}}$, $\mathcal{R}_{c_{n+1}}(n+1)$, and of c_{n+1} itself. Clearly, $\mathcal{R}_{c_{n+1}}(n+1) = \emptyset$, therefore, $\mathcal{I}_{c_{n+1}}(n+1)$ is the union of the edges in the route of c_{n+1} and the intervals $\mathcal{I}_{c_k}(n)$, for $c_k \in \mathcal{P}_{c_{n+1}}$. By the induction hypothesis, every $\mathcal{I}_{c_k}(n)$, for $c_k \in \mathcal{P}_{c_{n+1}}$, is an interval. Since every such interval $\mathcal{I}_{c_k}(n)$ intersects the route of c_{n+1}, then $\mathcal{I}_{c_{n+1}}(n+1)$ is an interval as well. We now need to show that $\mathcal{I}_{c_{n+1}}(n+1)$ is contained in $[y - x\beta, y + x + x\beta]$. Clearly, the route of c_{n+1} is contained in $[y - x\beta, y + x + x\beta]$. For any $c_k \in \mathcal{P}_{c_{n+1}}$ we showed above that $\mathcal{I}_{c_k}(n)$ is contained in $[y - x\beta, y + x + x\beta]$. Therefore, the interval $\mathcal{I}_{c_{n+1}}(n+1)$ is contained in $[y - x\beta, y + x + x\beta]$.

This completes the three possible cases for a call c_{n+1}. We showed that in each case the inductive hypothesis holds, and therefore it holds in general, which completes the proof of the theorem. □

Regarding optimality for this algorithm, we first note that (the adversary's behavior of) Theorem 4.2 can be easily adapted to hold for this model of call values given by the length of the routes. This yields a $\frac{1}{4}$ ($> \frac{1}{2g+1}$) competitiveness lower bound for this model. More recently, Furst and Tomkins have been able to close the gap, by providing a matching $\frac{1}{2g+1}$ lower bound [10].

3 Constant Call Value

In this section we consider the uniform value criterion, that is, a call value that is independent of the call's length and duration. The underlying network model and assumptions are the same as in the previous section (a line of nodes of size n), but now every call carries the same constant value, i.e., $\texttt{Val}(c_k) = a$ for all c_k and some $a > 0$. Without loss of generality, we will assume that $a = 1$. We call the on-line algorithm we present in this section `CV` , for *constant value.* Intuitively, since each call carries the same value, small calls (i.e., calls occupying only a few links) should be accepted, because they will leave room available for other calls, while big calls (calls spanning several links) should not. This is basically what `CV` does. Specifically, `CV` compares the size of the interval that each incoming call c_k requests, $|r_k|$, with the interval size of each of the calls that are required to be preempted in order to accommodate c_k (i.e., $|r_i|$ such that $c_i \in \mathcal{P}_{c_k}$). If $|r_k|$ is less than half of $|r_i|$, for each $c_i \in \mathcal{P}_{c_k}$, then c_k is accepted, otherwise it is discarded. That is, in the framework of Figure 1, we set $\alpha = -1$, $\beta = -\frac{1}{2}$, $f(c_k) = |r_k|$, and $h = \min_{c_i \in \mathcal{P}_{c_k}}\{|r_i|\}$.

The competitive factor of the algorithm we present is not a constant, but depends instead on the size of the network, although this dependence is only logarithmic. However, it turns out that this factor is optimal, i.e., no on-line algorithm for this model can compete better than ours.

We first give some definitions and prove some technical lemmas. As before, we resort to the transitive nature of preemption. We say that a call c_k *transitively preempts* a call c_0 if there exist calls $c_1, \ldots, c_{k-1}$ such that call c_i preempts call c_{i-1}, for $1 \leq i \leq k$. Note that for every call c that is preempted, there exists a call c' that transitively preempted c and was completed. We call c' the *root* of c.

Lemma 3.1 *Let c be a call that was preempted according to algorithm* `CV` *, and c' its root. Then the distance between the end points of c and c' is bounded by $|r_c|$.*

Proof: Denote by $c' = c_k, \ldots, c_0 = c$ the calls in the chain of preemption from c' to c. Algorithm CV guarantees that the length of c_{i+1} is less than half the length of c_i, i.e., $|r_i|/2 > |r_{i+1}|$. Therefore, the sum

$$\sum_{i=1}^{k-1} |r_i| < \sum_{i=1}^{k-1} 2^{-i}|r_0| < |r_0| \; .$$

To complete the proof, note that since it is the case that c_{i+1} preempted c_i, their intervals must have intersected (the distance between them is 0). Thus, the distance between the end points of c and c' is bounded by the above sum. □

Lemma 3.2 *Let c be a call that was rejected due to c'. Then the distance between the end points of c and the root of c' is bounded by $4|r_c|$.*

Proof: Since c was rejected because of c', $|r_{c'}| \leq 2|r_c|$. By Lemma 3.1 the distance between c' and its root is less than $|r_{c'}|$, therefore the distance between c and the root of c' is less than $2|r_{c'}|$, which in turn is bounded by $4|r_c|$.

□

We are now ready to establish the competitiveness of CV .

Theorem 3.3 *Algorithm* CV *is $\frac{a}{\log n}$-competitive, for some constant a.*

Proof: We would like to show that for any call algorithm CV completes, the off-line adversary can complete at most $O(\log n)$ calls. This would establish our theorem.

Consider a call c_k that is completed by CV , and all the calls that it caused to be preempted or rejected (either directly or transitively). By definition, the completed call is the root of all the calls in this set. Now consider the calls that the adversary could have scheduled from this set of calls. Naturally, these calls are non-overlapping. By Lemmas 3.1 and 3.2, the gap between the end points of each call c_i (preempted or rejected) in the set and c_k, the completed call, is less than four times the number of edges in c_i.

In general, calls in the set can be to the right or to the left of c_k. Consider first the calls on just one direction, say to the right. The situation above can be formalized as the following game, which bounds the adversary's strategy. Given the interval $[0, n]$, How many subintervals can we fit in it such that the distance between each subinterval and its root is at most 4 times its length, and the size of each subinterval is at least one? Note that the adversary's assignment obeys these criteria. (Also note that nothing can be gained by leaving gaps between the subintervals, since then one of the subintervals could be enlarged, without adding any new conflicts.) Thus, we have points $x_0, \ldots, x_l$ satisfying $0 \leq x_0 < \cdots < x_l$, $x_{i+1} - x_i \geq 1$, and $x_i \leq 4 \cdot (x_{i+1} - x_i)$.

From the last inequality we get that $(5/4) \cdot x_i \leq x_{i+1}$, which implies $(5/4)^l \cdot x_1 \leq x_l$. In addition, $x_1 \geq 1$, since each subinterval is of size at least one, and $x_l \leq n$, since it has to fit in the interval $[0, n]$. This implies $(5/4)^l \leq n$, which in turn means that $l \leq \log n / \log \frac{5}{4}$.

We view x_0 as the root, and can grow such a sequence in both directions. This implies that for any call that CV has completed the adversary has completed at most $2 \log n / \log \frac{5}{4} = O(\log n)$.

Any set of calls that the adversary would choose would have to obey the above rule of the game. That is, for any sequence of X calls that the adversary would produce, algorithm CV will serve at least $X \frac{a}{\log n}$ calls, for some constant a, which yields the theorem. □

We now show that CV is optimal for this model, by providing a matching lower bound for the competitive ratio. We first prove two technical lemmas. The first one states that we can assume for our purposes that an on-line algorithm always preempts an existing call whenever a shorter, overlapping call arrives.

Lemma 3.4 *Let c be a call being served, and c' an incoming call such that $r_{c'} \subset r_c$. For any on-line algorithm that rejects c' there is an online algorithm that preempts c and accepts c' and is at least as competitive.*

Proof: Consider any on-line algorithm that does not behave according to the statement of the lemma, that is, it does not always preempt an existing call whenever a shorter, overlapping call arrives. Consider the last time in a given sequence of calls in which the algorithm does it. We introduce a modification, by preempting the call being served and scheduling the incoming call instead. Note that with the modification the request sequence remains valid (i.e., we did not introduce any new conflicts), and that the new cumulative value remains the same. The first claim is true, since only a subset of the links used by the original algorithm are used now; the second claim holds since we deleted one call and added another call, and all calls are of the same value. Since we have not introduced any new conflicts, the above will remain true until the final state. We finish the proof by iteratively applying the above reasoning to the modified sequence. In the final sequence a shorter, overlapping call, is always accepted, and the value of the shorted call is no worse than the original sequence's value, yielding the lemma. □

The next lemma shows that we can assume, without loss of generality, that an on-line algorithm will accept an incoming call that does not overlap with any existing call.

Lemma 3.5 *An on-line algorithm that accepts an incoming call that does not overlap with any existing call is at least as competitive as an algorithm that rejects it.*

The proof of the lemma follows immediately, since the algorithm can always preempt the call under consideration in the future. We now prove that, for any on-line algorithm for this model, there exists a call sequence for which the algorithm is able to complete just one call, while the off-line algorithm completes $\Omega(\log n)$ calls. This makes algorithm CV optimal.

Theorem 3.6 *In the line model, when the value of the calls is constant, any on-line call control algorithm has a competitive factor of at most* $\frac{1}{\log n}$.

Proof: Consider an on-line algorithm that, without loss of generality, behaves according to Lemmas 3.4 and 3.5. Let the number of the nodes in the network be a power of 2, i.e., $n = 2^k$, for some k, and denote the nodes by $1, \cdots, n$. The adversary generates the following sequence: It starts with two calls, one from 1 to $n/2+1$ and the other from $n/2$ to n. The two calls intersect on the link $(n/2, n/2+1)$, and therefore only one of them can be accepted. (By Lemma 3.5 at least one of them is accepted.) Without loss of generality, assume that the online algorithm accepts the call from 1 to $n/2+1$. The adversary then accepts the call from $n/2$ to n, and continues recursively to generate calls in the interval 1 to $n/2$. Note that this interval does not intersect with the call that the adversary has accepted. Lemma 3.4 guarantees that when new calls appear that are subintervals of $(1, n/2+1)$, which is currently occupied by the call the on-line algorithm is running, then the algorithm will preempt this call, and continue with the new calls. The recursion ends with calls that require two links.

Hence, we have shown that for any on-line algorithm, for a network of size 2^k, there is a sequence of calls in which the algorithm completes only one call, while the adversary is able to complete k. This completes the proof of the theorem. $\square$

4 Call Value is Holding Time

In this section the abstract scenario is that of two nodes connected by a communication line such that no two calls can be accommodated at the same time. The calls' value is given by their holding time, i.e., $\texttt{Val}(c_i) = d_i$, for each call c_i. Again, a call that is prematurely terminated yields no value. In this context, $\mathcal{P}_{c_{new}} = \{c_{old}\}$, the existing call. We set $\alpha = 1$, $\beta = 2$ and $f = g = \texttt{Val}$. section and Namely, the algorithm—which we call `HT` , for *holding time*—accepts an incoming call only if its (estimated) holding time is more than twice the holding time of the existing call. (The parameters used for this case gives an algorithm very similar to the one designed independently for a certain scheduling problem in [7].) We remark that our analysis applies to time intervals of finite duration, since no on-line algorithm can guarantee a competitive factor with respect to time intervals of infinite duration.

Theorem 4.1 *Algorithm* `HT` *is* $\frac{1}{4}$*-competitive.*

Proof: Given a finite time interval, consider all the calls that were completed by `HT` . For each such call c_k, with, say, $d_k = l$, let again $\mathcal{P}^*_{c_k}$ be the transitive closure of $\mathcal{P}_{c_k}$, that is, the set of calls that c_k preempted, either directly or indirectly (i.e., by a call already in $\mathcal{P}^*_{c_k}$), and let $\mathcal{I}_{c_k}$ be the time interval that is the "union" of the time intervals corresponding to all the calls in $\mathcal{P}^*_{c_k}$. The following observations are used:

1. $|\mathcal{I}_{c_k}| < l \cdot \sum_{i=1}^{k-1} \frac{1}{2^i} < l$;

2. HT will reject any additional call c_{k+1} of duration up to $2l$.

An off-line algorithm could have accepted a sequence of "short," non-overlapping calls that can be superimposed to the above-described sequence, thereby covering the whole interval $\mathcal{I}_{c_k}$ (whose duration is at most l), the call c_k (whose duration is l), and the calls that c_k cause to reject (whose duration is at most $2l$). This implies that while HT has accrued l, the offline may accrue at most $4l$, which completes the proof of the theorem. □

The next theorem shows that the performance achieved by HT is optimal.

Theorem 4.2 *When the call value is given by the call's holding time, there does not exist an on-line call control algorithm with a competitive factor greater than* $\frac{1}{4}$.

The theorem follows from the lower bound derived by Baruah *et al.* (cf. Lemma 1, [7]) for the related on-line task scheduling problem in a uniprocessor environment. In such a setting, tasks requests arrive with an associated execution time and no slack time (i.e., the time between a task's arrival time and its deadline corresponds exactly to its computation time). Failure to allocate the processor to the task—due to rejection, or preemption by another, later-arriving task—results in a value of zero. The correspondence between this problem and call control with known holding time on a single link is immediate. Baruah *et al.* show that there does not exist an on-line scheduling algorithm with a competitive factor greater than 0.25. We refer the reader to [7] for further details.

5 Final Remarks

In this paper we have studied the problem of preemptive call control in an on-line fashion for a variety of call value criteria. We have provided algorithms that are competitive; furthermore, these algorithms are shown to be optimal.

In the models of Sections 2 and 3, it is assumed that both the holding time and the arrival time of all the calls are the same. It would be interesting to analyze the cases of same holding time, but arbitrary arrival times, as well as arbitrary holding times. The algorithms we present here are deterministic. What happens in the length-of-route and constant-value models when randomization is allowed?

Acknowledgements

The work of Juan Garay was partly done while the author was visiting the Centrum voor Wiskunde en Informatica (CWI) in Amsterdam. The author is thankful to the Center for its hospitality. The authors would also like to thank the two referees for their attentive reading of a preliminary version of the manuscript, and for pointing out—and suggesting fixes for–several inconsistencies.

References

[1] B. Awerbuch, Y. Azar, and S. Plotkin, "Throughput Competitive On-Line Routing," *Proc 34th IEEE Annual Symp. on Foundations of Computer Science*, pp. 32-40, Palo Alto, CA, November 1993.

[2] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts, "Competitive Routing of Virtual Circuits with Unknown Duration," *Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 321-327, Arlington, VA, January 1994.

[3] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén, "Competitive Non-Preemptive Call Control," *Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms*, pp. 312-320, Arlington, VA, January 1994.

[4] B. Awerbuch, R. Gawlick, T. Leighton, and Y. Rabani, "Online Admission Control and Circuit Routing for High Performance Computing and Communication," *Proc 35th IEEE Annual Symp. on Foundations of Computer Science*, pp. 412-423, Santa Fe, NM, November 1994.

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

[6] A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour and B. Schieber, "Bandwidth Allocation with Preemption," to appear in *Proc. STOC 1995.*

[7] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha and F. Wang, "On the Competitiveness of On-Line Real-Time Task Scheduling," *Proc. IEEE Real-Time Systems Symposium*, pp. 106-115, 1991.

[8] D. Clark, B. Davie, D. Farber, I. Gopal, B. Kadaba, W. Sincoskie, J. Smith and D. Tennenhouse, "An Overview of the AURORA Gigabit Testbed," *Proc. INFOCOM '92*, Florence, Italy, pp. 569-581, 1992.

[9] I. Cidon and I. Gopal, "PARIS: An approach to integrated high-speed private networks," *Int. Journal of Digital and Analog Cabled Systems,* 1(2), pp. 77-86, 1988.

[10] M. Furst and A. Tomkins, "Some Lower Bounds for Call Control Algorithms," private communication.

[11] J.A. Garay and I.S. Gopal, "Call Preemption in Communication Networks," *Proc. INFOCOM '92*, Florence, Italy, pp. 1043-1050, 1992.

[12] S. Plotkin, "Competitive Routing of Virtual Circuits in ATM Networks," invited paper, *IEEE J. Selected Areas in Communications.* `http://theory.stanford.edu/people/plotkin/routing.html`.

[13] D. Sleator and R. Tarjan, "Amortized efficiency of list update and paging rules," *Communications of the ACM,* 28(2), pp. 202-208, 1985,

[14] J.S. Turner, "New Directions in Communications (or Which Way to the Information Age?)," *IEEE Commun. Mag.*, Vol. 24, pp. 8-15, Oct. 1986.