On-line Randomized Call Control Revisited*

Stefano Leonardi[††] Alberto Marchetti-Spaccamela[‡] Alessio Presciutti[†]
Adi Rosén[§]

Abstract

We consider the on-line problem of call admission and routing on trees and meshes. Previous work considered randomized algorithms and analyzed the *competitive ratio* of the algorithms. However, these previous algorithms could obtain very low profit with high probability.

We investigate the question if it is possible to devise on-line competitive algorithms for these problems that would guarantee a "good" solution with "good" probability. We give a new family of randomized algorithms with provably optimal (up to constant factors) competitive ratios, and provably good probability to get a profit close to the expectation. We also give lower bounds that show bounds on how high the probability of such algorithms, to get a profit close to the expectation, can be.

We also hope that this work could serve as a first step towards understanding how well can the profit of an competitively-optimal randomized on-line algorithm be concentrated around its expectation.

1 Introduction

The area of communication networks gives rise to a large number of on-line problems. This is because network resources are often limited, while the network traffic is often unpredictable. One of the most extensively studied problems in this area is the problem of on-line assignment of virtual circuit in networks: a sequence of requests for calls is given on-line to an algorithm which has to assign a virtual circuit between the communicating parties (or to reject the request), obeying the network constraints (such as link capacities).

Two kind of decisions are involved. *Admission control*, i.e., the choice whether to accept the call or not, and *route selection*, i.e., the decision on the route of an accepted call. We consider the case in which these two decisions are taken in an on-line fashion. Each call is processed before any future call is known, while the algorithm has to take both decisions. The appeal of this problem stems also

*An extended abstract of this work will appear in the Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms (SODA '98)

†Max-Planck Institute für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany.

‡Dipartimento di Informatica Sistemistica, Università di Roma "La Sapienza", via Salaria 113, 00198-Roma, Italia.
This work was partly supported by EU ESPRIT Long term Research Project ALCOM-IT under contract n. 20244, and by Italian Ministry of Scientific Research Project 40% "Algoritmi, Modelli di Calcolo e Strutture Informative".
e-mail: {leon,marchetti,presciutti}@dis.uniroma1.it

§Department of Computer Science, University of Toronto, Toronto, Canada. email: adiro@cs.toronto.edu

from the fact that in its basic form (when link capacities are 1, and each call requests bandwidth 1), it is an on-line version of the extensively studied combinatorial problem of the maximization of the number of edge-disjoint paths. The aim of the on-line algorithm is to maximize the *benefit*, in our case usually the number of accepted calls.

Algorithms for such on-line maximization problems are usually analyzed by their *competitive ratio* [ST85], the worst case, over all input sequences, of the ratio between the benefit obtainable by an off-line optimal algorithm that knows the whole sequence in advance, and the benefit obtained by the on-line algorithm.

As greedy strategies for problems of the kind we consider here are clearly undesirable (as a call may require a lot of resources, and then prevent accepting other calls that would yield better benefit) more involved strategies are necessary. Awerbuch, Azar and Plotkin [AAP93] proposed a deterministic on-line algorithm (denoted AAP in this paper) for the case in which the transmission rate (bandwidth request) of each call is small with respect to the link capacities of the network. Their algorithm works for any network topology, and achieves a benefit that is at least a logarithmic (in the size of the network) fraction of the optimal solution. No deterministic algorithm can have an asymptotically better competitive ratio. However, if calls may request the full bandwidth of links in the network, [AAP93] prove a lower bound of $\Omega(n)$ for deterministic algorithms for a line network, where n is the number of vertices of the network. (Deterministic algorithms with good competitive ratio are still possible for the edge disjoint paths problem on some particular networks as expander graphs [KR96].)

Therefore, a lot of attention in research has been given to the study of *randomized* on-line algorithms for the problems we consider. Also these randomized algorithms are analyzed again using their competitive ratio. Such an algorithm, A, is said to be c-competitive [1] if for *any* sequence of requests σ, $E[Benefit_A(\sigma)] \geq \frac{1}{c} \cdot Benefit_{OPT}(\sigma)$, where the expectation is taken over the random choices of the algorithm.

Several results have been obtained in the design of randomized on-line algorithms with good competitive ratios for some important network topologies. An algorithm with $O(\log n)$ competitive ratio has been proposed for a line and a tree network of n vertices by Awerbuch, Bartal, Fiat and Rosén [ABFR94]. This result has been improved to $O(\log D)$, where D is the diameter of the tree, by Awerbuch, Gawlick, Leighton and Rabani [AGLR94]. Kleinberg and Tardos [KT95] presented an algorithm with $O(\log n)$ competitive ratio for meshes, that is extended to a class of planar graphs. For all these network topologies, matching (up to a constant factor) randomized lower bounds have also been proved. On the other hand, Bartal, Fiat and Leonardi [BFL96] prove an $\Omega(n^\epsilon)$ lower bound for randomized on-line algorithms on a specific network.

However, all the above mentioned algorithms only have the property that their *competitive ratio* is "good". They have not been analyzed with respect to their probability to get a "good" solution; their guarantee is that for any sequence the *expected* benefits is "close" (usually a logarithmic fraction) to the optimum solution. It may very well be that with high probability a very poor benefit is achieved (and the main contribution to the average is given by a very high benefit, obtained with low probability). Indeed, algorithms of this kind have been criticized on this basis as not giving an appropriate solution for the problems they attempt to solve.

[1] against an oblivious adversary [BBKTW90].

For instance, the $O(\log n)$-competitive algorithm of [ABFR94] for maximum edge-disjoint paths on trees obtains for some sequences, even with $OPT = \Omega(n)$, a benefit of 0 with probability $1-\frac{1}{\log n}$. The $O(\log D)$ algorithm for trees [AGLR94] behaves a somewhat better; still *unless* $OPT = \Omega(n^{\epsilon})$, the probability to get any constant fraction of the expectation remains $o(1)$ (as a function of n). As to meshes, the algorithm of [KT95] decides with probability 1/2 to accept only "short" calls and with probability 1/2 to accept only "long" calls. On sequences that are composed only from one kind of calls, the algorithm would obtain benefit 0 with probability at least 1/2.

In this paper we investigate the question whether it is possible to obtain randomized on-line algorithms (for the problems at hand) which would guarantee with "good" probability a "good" solution. The probability to achieve a benefit close to the expectation is actually intimately related to the quality of the expectation. Obviously, the expectation of deterministic algorithms, that have very poor competitive ratios for these problems, is obtained with probability 1.

We identify and formulate the following question. Our results give answers to most of them.

(1) Can a randomized on-line algorithm with optimal (up to a constant factor) competitive ratio, guarantee a benefit of a constant fraction of the expectation with *constant* probability ?

(2) Can a randomized on-line algorithm with optimal competitive ratio achieve a constant fraction of the expectation with probability that tends to 1? (Say, as the size of the optimal solution grows.)

(3) What is the interplay between the competitive ratio of a randomized algorithm, and the concentration of the solution around the expectation ?

1.1 Results of the paper

We address the above questions for the call admission problem, in the basic case of infinite duration, request bandwidth and link capacities 1, on trees and meshes. This problem reduces to the on-line version of the maximum edge-disjoint path problem on these topologies.

We present new randomized algorithms for trees and meshes with provably good competitive ratio and provably good concentration around the expected benefit.

Our work contains the following results for trees:

- We present randomized algorithms with competitive ratios of $k \log 2D$, $k \geq 16$, with the guarantee that the probability to obtain a $1-\delta$ fraction of the expectation, $\delta \in [0,1]$, tends to a value $P(k,\delta)$, as $\frac{OPT}{\log D}$ tends to infinity. The value $P(k,\delta)$ is a constant bounded away from 1, if k is a constant, but it can be made as close to 1 as desired by making k larger. The rate of convergence is such that already for sequences with $OPT = \Omega(\log D)$ our algorithms with competitive ratio $O(\log D)$ achieve any constant fraction of the expectation with probability $\Omega(1)$.
- We prove that no algorithm with $O(\log D)$ competitive ratio can obtain a constant fraction of the expectation with probability that tends to 1, even as the size of the optimal solution grows (with respect to $\log D$). We thus prove that our algorithms have the best possible behavior around the expectation for optimal (in terms of competitive ratio) algorithms.

- We also show that other members of our family of algorithms with slightly worse competitive ratio of $O(\log^{1+\epsilon} D)$, for arbitrary $\epsilon > 0$, have the property to get a constant fraction of the expectation with probability that tends to 1, as $\frac{OPT}{\log^{1+\epsilon} D}$ tends to infinity.
- Our family of algorithms for trees is based on a new and simple design. We are also able to derive an algorithm with the best known competitive ratio for the problem, $4 \log 4D$, as opposed to the previously known ratio of $48 \log 2D$ of [AGLR94]. Finally, we show that our algorithms apply to trees with any uniform integer capacities for the edges. In fact, we show that *any* algorithm for trees of link capacities 1 can be converted to the case of arbitrary uniform capacities, with almost the same competitive ratio.

Later we study the problem on meshes:

- We present an optimal $O(\log n)$ competitive algorithm for meshes that obtains a constant fraction of the expectation with probability that tends to 1, as $\frac{OPT}{\log^4 n}$ tends to infinity. This algorithm is based on some of the ideas of [KT95] and [BL97] and on new ideas presented here.

Beyond our results on this specific set of problems, we hope that this work could be a first step towards understanding the relationship between competitiveness and the concentration of the solution around the expected benefit for on-line randomized algorithms for benefit problems.

2 The algorithms for trees

In this section we describe new randomized on-line algorithms for call admission on trees. We design a family of randomized algorithms with different competitive ratios, and analyze how their performance concentrates around the expectation.

Our algorithms follow a new approach. They are composed of two conceptual steps. In the first step we apply an on-line *deterministic filter* to the input sequence. Those calls that are *not* filtered out are called *candidate calls*, and may be accepted. The set of the candidate calls has two important properties: firstly, its cardinality is a constant fraction of the cardinality of the largest (optimal) set of calls that can be accepted (by an off-line adversary). Secondly, although this set cannot be fully accepted (as some of the calls intersect), the intersections between the calls of this set exhibit "nice" properties. When a call passes the deterministic filter and becomes a candidate call it is presented to an on-line *randomized selection procedure* that determines if the call is actually accepted. The randomized selection procedures that we use are very simple, and the various members of the family are distinguished by the randomized selection procedure used.

Before we describe the algorithms, we give some notations: We denote by $\sigma = (s_1, t_1), (s_2, t_2), \ldots$ the sequence of requests. We denote by $OPT(\sigma)$ and $\mathcal{A}(\sigma)$ the set of calls accepted by an optimal (off-line) algorithm, and by an on-line algorithm $\mathcal{A}$, respectively, out of the sequence σ. We denote by $\mathcal{C}(\sigma)$ the set of candidate calls, passed on by the deterministic filtering procedure to the randomized selection procedure. D indicates the diameter of the tree.

We start by describing the properties of the deterministic filter that we use.

Theorem 1 *There exists a deterministic on-line filtering procedure such that for any sequence σ the following properties hold:*

- $|\mathcal{C}(\sigma)| \geq |OPT(\sigma)|/4$.
- *The number of pairs of calls in $\mathcal{C}(\sigma)$ that intersect is at most $|\mathcal{C}(\sigma)| \cdot \log 2D$. (In other words, the number of edges in the intersection graph of the calls in $\mathcal{C}$ is at most $|\mathcal{C}(\sigma)| \cdot \log 2D$).* [2]

We prove this theorem in Section 2.2. We also show in Section 2.3 an alternative filtering procedure which makes use of the AAP algorithm, but achieves somewhat worse performances.

The different members of the family of algorithms are distinguished by the randomized selection procedure used. Actually, we use a single procedure parameterized by a parameter p. Algorithm $\mathcal{A}_p$ presents the candidate calls selected by the deterministic filter, one by one as they are selected, to the on-line randomized selection procedure $\mathbf{RS}_p$, defined below.

$\mathbf{RS}_p$: Any candidate call $(s,t) \in \mathcal{C}$ becomes a *considered* call with probability p, and is rejected with probability $1-p$. A considered call is accepted if it does not intersect any previous *considered* call. [3]

Clearly, no two calls that are accepted by the above procedure intersect, and therefore they can all be accepted.

2.1 Analysis of the algorithms

We prove the competitive ratio of our algorithms and the statement on the deviation from the expectation.

Theorem 2 *Algorithm $\mathcal{A}_p$ is a $\frac{4}{p(1-p\log 2D)}$-competitive randomized on-line algorithm for call admission on trees of diameter D. For any $\delta \in [0,1]$, and any sequence σ, $Pr[\mathcal{A}_p(\sigma) < (1-\delta)E[\mathcal{A}_p(\sigma)]] \leq exp(-\frac{|OPT(\sigma)|p}{8}(\frac{1}{2}\delta(1-p\log 2D))^2) + \frac{1}{1+\frac{1}{2}\delta(\frac{1}{p\log 2D}-1)}$.*

Before proving the theorem we explain its consequences and give some corollaries.

Corollary 3 *For any constant $k \geq 16$, algorithm $\mathcal{A}_p$ with $p = \frac{1}{2\log 2D} \cdot (1-\sqrt{1-(16/k)})$ is a $k\log 2D$-competitive randomized algorithm for on-line call admission on trees of diameter D. The probability that $|\mathcal{A}_p(\sigma)| \geq (1-\delta)E[\mathcal{A}_p(\sigma)]$ tends to $P(k,\delta) = 1 - \frac{1}{1+\frac{1}{2}\delta(\frac{1}{p\log 2D}-1)} = 1 - \frac{1}{1+\frac{1}{2}\delta(2\frac{1}{1-\sqrt{1-\frac{16}{k}}}-1)}$ as $|OPT(\sigma)|/\log D$ tends to infinity. In particular for every sequence σ with $|OPT(\sigma)| = \Omega(\log D)$ the probability to get a $(1-\delta)$ fraction of the expectation is $\Omega(1)$.* [4]

[2] The intersection graph is a graph $G=(V,E)$ where each node in V corresponds to a call in $\mathcal{C}$, and there is an edge $(v,u) \in E$ if and only if the calls that correspond to v and u intersect.

[3] The more natural algorithm that immediately rejects a candidate call if it intersects a previous *accepted* call, and otherwise accepts the candidate with probability p, can be easily shown to perform at least as well. We leave the description as above for simplicity of the proof.

[4] To see that, note that $p = \Theta(1/(2\log 2D))$, thus the absolute value of the expression in the exponent of the first term is at least some constant times $|OPT(\sigma)|/\log 2D$.

Observe that the value $P(k,\delta)$, for constant k, is a constant bounded away from 1, but it can be made as close to 1 as desired, at the expense of a larger k. In this case also the competitive ratio is increased by a constant multiplicative factor.

Corollary 4 *Algorithm $\mathcal{A}_p$, with $p=\Theta(\frac{1}{\log^{1+\epsilon} D})$, $\epsilon>0$, is an $O(\log^{1+\epsilon} D)$-competitive randomized algorithm for on-line call admission on trees of diameter D. The probability that $|\mathcal{A}_p(\sigma)| \geq (1-\delta)E[\mathcal{A}_p(\sigma)]$ tends to 1 as $|OPT(\sigma)|/\log^{1+\epsilon} D$ tends to infinity.*

As to the best competitive ratio achievable by this family of algorithms, we observe that by optimizing the expression $\frac{4}{p(1-p\log 2D)}$ we can get competitive ratio of $16\log 2D$.

Proof of Theorem 2. We prove that the expected number calls accepted by $\mathcal{A}_p$ on a sequence σ, is at least $|\mathcal{C}(\sigma)|\cdot p(1-p\log 2D)$, where $\mathcal{C}(\sigma)$ is the set of candidate calls picked from σ. By Theorem 1 $|\mathcal{C}(\sigma)| \geq |OPT(\sigma)|/4$, and the competitive ratio follows.

We number the *candidate* calls by the order that they arrive. Let ℓ be the number of candidate calls, that is, $\ell = |\mathcal{C}(\sigma)|$. If calls i and j intersect we denote this by $i\cap j$. Let $c_i, i=1,\ldots,\ell$ be an indicator random variable which is 1 if the ith candidate is *considered*, and 0 otherwise. Let $C=\sum_{i=1}^{\ell} c_i$. Let $y_{i,j}$, for $i<j$ and $i\cap j$, be an indicator random variable which is 1 if both calls i and j are considered, and 0 otherwise, and let $Y=\sum_{i<j, i\cap j} y_{i,j}$. Let r_j be an indicator random variable which is 1 if there is an index $i<j$, $i\cap j$, such that $y_{i,j}=1$. Let $R=\sum_{j=1}^{\ell} r_j$. Note that $R\leq Y$. In the following we abuse notation and denote $|\mathcal{C}(\sigma)|$ by simply $\mathcal{C}$.

A candidate is accepted if it was considered, but none of its (previously presented) intersecting candidates were considered. Thus, if candidate j is considered but not accepted into the on-line solution, then there is a candidate $i<j$, $i\cap j$, that was considered. That is, this situation corresponds to the case where $c_j=1$, and $r_j=1$. Therefore $|\mathcal{A}_p(\sigma)| = C-R \geq C-Y$, and we get $E[|\mathcal{A}_p(\sigma)|] \geq E[C]-E[Y]$.

Clearly $E[C]=\mathcal{C}\cdot p$ since each candidate is considered with probability p. For any pair $i<j$, $i\cap j$, $Pr[y_{i,j}=1]=p^2$ since the event occurs if and only if call i and call j are considered. By Theorem 1 there are at most $\mathcal{C}\cdot\log 2D$ pairs $i<j$, such that $i\cap j$. Therefore $E[Y]\leq \mathcal{C}\log 2D\cdot p^2$. We get that $E[|\mathcal{A}_p(\sigma)|] \geq E[C]-E[Y] \geq \mathcal{C}(p-\log 2D\cdot p^2) = \mathcal{C}\cdot p(1-p\log 2D)$.

We now turn to prove a bound on the probability that the algorithm achieves a profit of less than $(1-\delta)$ times the expectation, for $\delta\in[0,1]$. Since $E[C]\geq \frac{E[Y]}{p\log 2D} \geq \frac{E[R]}{p\log 2D}$, we have:

$$\begin{aligned}
Pr[|\mathcal{A}_p(\sigma)| < (1-\delta)E[|\mathcal{A}_p(\sigma)|]] &= Pr[(C-R) < (1-\delta)E[C-R]] \\
&\leq Pr[C < E[C] - \frac{1}{2}\delta E[C-R]] + \\
&\quad\; Pr[R > E(R) + \frac{1}{2}\delta E(C-R)] \\
&\leq Pr[C < E[C] - \frac{1}{2}\delta(1-p\log 2D)E[C]] + \\
&\quad\; Pr[R > E[R] + \frac{1}{2}\delta(\frac{1}{p\log 2D}-1)E[R]] \\
&= Pr[C < (1-\frac{1}{2}\delta(1-p\log 2D))\cdot E[C]] +
\end{aligned}$$

$$Pr[R > (1 + \frac{1}{2}\delta(\frac{1}{p\log 2D} - 1)) \cdot E[R]] \ .$$

The variable C is a sum of $\mathcal{C}$ random variables set by independent Bernoulli trials; the variable R is also a sum of random variables, but they are not independent since for a fixed i, for all pairs $i, j, i < j$, $y_{i,j}$ depends upon the choice for candidate i. We use Chernoff's bounds to bound the first term and the Markov inequality for the second term:

$$\begin{aligned} Pr[|\mathcal{A}_p(\sigma)| < (1-\delta)E[|\mathcal{A}_p(\sigma)|]] &< exp(-\frac{E[C]}{2}(\frac{1}{2}\delta(1 - p\log 2D))^2) + \frac{1}{1 + \frac{1}{2}\delta(\frac{1}{p\log 2D} - 1)} \\ &= exp(-\frac{\mathcal{C}p}{2}(\frac{1}{2}\delta(1 - p\log 2D))^2) + \frac{1}{1 + \frac{1}{2}\delta(\frac{1}{p\log 2D} - 1)} \\ &\leq exp(-\frac{|OPT(\sigma)|p}{8}(\frac{1}{2}\delta(1 - p\log 2D))^2) + \frac{1}{1 + \frac{1}{2}\delta(\frac{1}{p\log 2D} - 1)} \ . \end{aligned}$$

□

2.2 The deterministic filter

When call (s,t) is presented to the algorithm, it is first passed through the deterministic filter, where it is discarded or becomes a candidate. We describe in the following a deterministic filter appropriately designed for trees.

For this deterministic filter we designate an arbitrary internal vertex as the root of the tree, and denote it by r (If there is no internal vertex, the tree is formed by a single edge and the problem becomes trivial). We restrict our description to calls between two leaves of the tree. This is general enough to capture the general problem: for each non-leaf vertex v, we can add additional leaves $l_{v,e}$, for each edge e incident to v. Each call that has an endpoint in v and includes edge e is then transformed to a call that has an endpoint in $l_{v,e}$. A pair of calls intersects in the new instance if and only if the corresponding pair intersects in the original instance. The diameter of the tree does not increase by this procedure. We denote by $lca(s,t)$ the *least common ancestor* of vertices s and t in the tree rooted at vertex r, and by $p(u)$ the parent of vertex u in the tree rooted at r. $path(s,t)$ is the *set* of edges in the path connecting s to t. Finally, $\mathcal{T}$ is the set of all edges of the tree.

Calls can be discarded on the basis of two tests. To perform the first test we block 2 edges when a call becomes a candidate: For any candidate (s',t') the two edges on $path(s',t')$ adjacent to $lca(s',t')$ are blocked. The edges are said to be *blocked edges.*

First test. Discard call (s,t) if $path(s,t)$ contains a blocked edge.

If call (s,t) is not discarded by the first test, then it is submitted to the second test, for which we associate a weight function $w(e)$ to each edge of the tree; the value is determined as follows:

1. Initially, assign $w(e) = 1$, for all edges $e \in \mathcal{T}$.

2. Double $w(e)$ whenever edge e is included in a $path(s',t')$ for a call (s',t') that becomes a candidate.

Second test Discard (s,t) if there is a weight w and a pair of vertices u, v on $path(lca(s,t),s)$ or on $path(lca(s,t),t)$, such that $w(e) = w$ for all $e \in path(u,v)$, and $|path(u,v)| \geq \frac{2D}{w}$.

We prove that the number of candidates, i.e., those calls that are not discarded by any of the two tests, is at least $OPT(\sigma)/4$. We need some intermediate steps. In the following we will often say that edge e is included in a call (s,t) if $e \in path(s,t)$.

Claim 5 *A candidate (s,t) only intersects previous candidates that include edge $(lca(s,t), p(lca(s,t)))$.*

Proof. If there is a previous candidate (s',t') that does not include edge $(lca(s,t), p(lca(s,t)))$, but intersects call (s,t), then $lca(s',t')$ is in the subtree rooted at $(lca(s,t))$. Then (s,t) intersects (s',t') in one of the two blocked edges of (s',t'), a contradiction to test 1. ■

Definition. Given two vertices (u,v), $path(u,v)$ is a *segment* if there exists a set of candidates $\mathcal{C}'$ such that $path(u,v) = \bigcap_{(s,t)\in\mathcal{C}'} path(s,t)$, and no edge of $path(u,v)$ is included in a candidate call not in $\mathcal{C}'$. We denote such a segment, and the its set of edges, by $seg(u,v)$.

Observe that the edges of a segment have equal weights. Also, any set of edges that is the intersection of a set of paths, necessarily forms a path.

Claim 6 *If the edges of a segment $seg(u,v)$ have weight $w > 2$ then either u is an ancestor of v or v is an ancestor of u.*

Proof. If $w > 2$ then the edges of the segment are included in more than one candidate. Let (s,t) be the first call. If neither u is an ancestor of v nor v is an ancestor of u, then the segment includes $lca(s,t)$; two edges of the segment would become blocked when (s,t) becomes a candidate, and the second call cannot become a candidate, by test 1. ■.

Claim 7 *Let (s,t) be a candidate. When (s,t) is presented, any $path(u,v)$ contained in $path(lca(s,t),s)$ or $path(lca(s,t),t)$ with all edges of equal weight w, is part of a segment of weight w.*

Proof. By Claim 5 all the previous candidates that intersects (s,t), include edge $(lca(s,t), p(lca(s,t)))$. Therefore, if we follow a path from $lca(s,t)$ to endpoint s or t, we observe that the weight is non increasing. The weight is decreased (at least halved) every time a previous candidate leaves the path to follow a different direction. Any path $path(u,v)$ formed by edges of equal weight is thus contained in the the intersection of a set $\mathcal{C}'$ of candidates, such that no edge of $path(u,v)$ is included in a candidate call not in $\mathcal{C}'$, and, hence, is contained in a segment. ■

The definition of segment is critical for our purposes, since the number of calls that are discarded by the deterministic filter at the second test, while accepted in the optimal solution, is strictly related to the maximum number of segments of weight bigger than 2 in the tree.

We prove the following upper bound on the number of such segments:

Lemma 8 *The number of segments of weight bigger than 2 is at most $2\mathcal{C}$.*

Proof. If a call is filtered out, then no new segment is created. We prove that when a call (s,t)

becomes a candidate, the number of segments of weight bigger than 2 is increased at most by 2. We distinguish two cases.

In the first case, all the intersecting previous candidates intersect (s,t) only on edges of $path(lca(s,t),s)$ (or equivalently $path(lca(s,t),t)$). In this case, $lca(s,t)$ can be in the middle of a segment $seg(u,v)$ and $path(lca(s,t),s)$ contains only a part of the edges of $seg(u,v)$. The segment is broken into two new segments, then increasing by 1 the number of segments of weight bigger than 2. The edges of $path(lca(s,t),t)$ were not part of any candidate, while now form a segment of weight 2.

By Lemma 6, we can assume that any segment $seg(u,v)$ of weight bigger than 2 has u ancestor of v. We now follow $path(lca(s,t),s)$ from $lca(s,t)$ to s, crossing a sequence of segments from the beginning to the end, unless we leave one, let us say again $seg(u,v)$, in the middle, at vertex u' before to reach v. This segment is broken into two segments, then increasing by 1 the number of segments of weight bigger than 2. However, the edges from u' to s were not part of any previous candidate. Otherwise there would have been a previous candidate crossing only a part of $seg(u,v)$, a contradiction. Now $seg(u',s)$ is a segment of weight only 2.

In the second case, previous intersecting candidates intersect edges of both $path(lca(s,t),s)$ and $path(lca(s,t),t)$. $lca(s,t)$ cannot be in the middle of a segment including edges of $path(s,t)$, and then no new segment is created there. On the other side, the number of segments of weight bigger than 2 can be increased by 1 for both $path(lca(s,t),s)$ and $path(lca(s,t),t)$, if they break in the middle a segment on the way from $lca(s,t)$ to s or t. ■

We will finally prove that the number of calls not discarded are a good fraction of the optimal solution.

Lemma 9 *The number of candidate calls is* $\mathcal{C} \geq \frac{OPT}{4}$.

Proof. We prove that the optimal solution can accept at most 4 calls for each candidate that is discarded by the deterministic filter. Calls are discarded at the first test, if they cross blocked edges. Since there are at most $2\mathcal{C}$ blocked edges in the tree, there are at most $2\mathcal{C}$ of these calls that are not candidated but accepted in an optimal solution, or that are candidated and also accepted in the optimal solution.

We are left to consider calls that are rejected at the second test but are accepted in an optimal solution. Consider a call (s,t) of such kind. It crosses, when presented, the edges of a path $path(u,v)$ of at least $\frac{2D}{w}$ edges of equal weight w on $path(lca(s,t),s)$ or $path(lca(s,t),t)$. By Lemma 7, $path(u,v)$ is part of a segment $seg(u',v')$ of weight $w > 2$. Observe that no call is discarded because of a path of weight 2 since $path(lca(s,t),s)$ and $path(lca(s,t),t))$ are of size at most $D-1$.

All the edges of this segment are crossed by the same candidated calls. The last of such candidates, say (s',t'), raised the weight of all the edges of $seg(u',v')$ from $w/2$ to w. (s',t') passed the second test, and then every path with edges of weight $w/2$ included in $path(s',t')$, is formed by less than $\frac{4D}{w}$ edges. This is an upper bound on the number of edges of $seg(u',v')$. A call is not candidated because of a $path(u,v)$ of weight w that includes at least $\frac{2D}{w}$ edges of $seg(u',v')$, more than 1/2 of the edges of the segment. It follows that at most one of all the calls that are discarded at the second test for a path of weight w formed by edges of $seg(u',v')$ can be accepted in an optimal solution. By Lemma 8, at most $2\mathcal{C}$ segments of weight bigger than 2 are created while the algorithm processes the sequence, and then at most $2\mathcal{C}$ calls in the optimal solution are discarded because of the second test. ■

Lemma 10 *Every candidate call intersects at most* $\log 2D$ previous *candidates.*

Proof. By Claim 5 we know that for any candidate (s,t), all the previous candidates that intersect it include edge $(lca(s,t), p(lca(s,t)))$. Their number is at most $\log 2D$, since once an edge is included in $\log 2D$ candidates, its weight is $2D$, and then no further call that includes this edge can become a candidate (by test 2). ■

We can now prove the properties of the filter.

Proof of Theorem 1. The first part of the theorem is proved in Lemma 9. For the second part of the theorem, by Lemma 10 every candidate call of $\mathcal{C}$ intersect at most $\log 2D$ previous candidate calls; this shows that the number of intersecting pairs is bounded by $|\mathcal{C}| \cdot \log 2D$. □

2.3 An alternative deterministic filter

In this section we describe an alternative design of the deterministic filter, at the cost of having a somewhat worse performance. This filter uses the AAP (deterministic) call control algorithm [AAP93], originally designed for general networks of high capacity.

The test runs the AAP algorithm on the tree, assuming link capacities of $\log 4D$. A call that is accepted by the AAP algorithm becomes a candidate, while a call that is rejected by AAP, is also rejected by the filter. This filter relies on the following property: if the capacity is increased in that way, then the set of calls accepted by AAP is a constant fraction of the set accepted by an adversary that has only capacity 1. (A similar result is also derived in [LM95], if the capacity is increased by a factor of $\Theta(\log n)$). On the other hand, since the capacity of each link is bounded, the number of intersections between the candidate calls is small.

For completeness we describe the AAP algorithm, for the above stated parameters. Let $u(e) = \log 4D$, $e \in E$ be the capacity of edge e of the tree. Let $b_j(e)$ be the number of calls routed trough edge e by AAP among the first j calls. Define the relative load of edge e, just after call j has been processed, to be $\lambda_j(e) = \frac{b_j(e)}{u(e)}$. Define the cost of edge e just after call j has been processed to be $c_j(e) = u(e)[\mu^{\lambda_j(e)} - 1]$, where $\mu = 4D$. A request (s_j, t_j) is accepted if $\sum_{e \in path(s_j,t_j)} \frac{1}{u(e)} \cdot c_{j-1}(e) \leq D$.

We now give the properties of the filter:

Lemma 11 *The above filter, applied to trees of diameter D, achieves for any sequence σ:*

- $|\mathcal{C}(\sigma)| \geq |OPT(\sigma)|/5$.
- *The number of pairs of calls in $\mathcal{C}(\sigma)$ that intersect is at most $|\mathcal{C}(\sigma)| \cdot 2\log 2D$. (In other words, the number of edges in the intersection graph [2] of the calls in $\mathcal{C}$ is at most $|\mathcal{C}(\sigma)| \cdot 2\log 2D$).*

Proof Sketch. The first part of the claim follows from two modifications of the proof of the original AAP algorithm [AAP93]. First, we observe that the parameter n (number of edges), can be changed to D (the length of the longest simple path) (This has already been observed in [AGLR94]). Then, we note that if the adversary is "impoverished" with respect to the AAP algorithm in that it has less capacity by a factor of $\log \mu = \log 4D$, then the last inequality in the proof can be modified, and shows that the competitive ratio becomes a constant (instead of $O(\log \mu)$). We omit the details from this abstract.

For the second part of the claim we consider an intesection graph of paths on a tree. Our claim follows from the fact that if each edge in the tree is included in at most g of these paths, then the intersection graph is a $(2(g-1))$-inductive graph (this is proved in [BL97]). [5] Since the capacity used by the AAP algorithm is $\log 4D$, any edge is used by at most $\log 4D$ calls, and the number of intersections is bounded by $|\mathcal{C}(\sigma)| \cdot 2(\log 4D - 1) = |\mathcal{C}(\sigma)| \cdot 2 \log 2D$. □

The randomized selection procedure used in conjunction with this filter is the same as for our first filter. The results stated in Theorem 2 are proved in a similar way, with the necessary modification in the constants, yielding somewhat larger best-possible competitive ratio, a different constant as the limit for the probability and a different rate of convergence.

3 A bound on the probability to achieve a "good" solution

We have presented $O(\log D)$ competitive algorithms for call admission on trees, that for sequences σ, with $OPT(\sigma) = \Omega(\log D)$, achieve any constant fraction of the expected benefit with at least constant probability. The obvious question is if this statement can be made stronger, i.e., if there exist on-line algorithms with optimal $(O(\log D))$ competitive ratios, that at the same time achieve a constant fraction of the expectation with probability that tends to 1 (as the optimal solution grows).

In this section we answer this question in the negative proving that for any algorithm with an $O(\log D)$ competitive ratio, there is a sequence σ with $OPT = \Theta(\log^{2/3} D)$, for which with probability $\Omega(1)$, the on-line benefit is $o(OPT/\log D)$ (in fact the on-line benefit is only a $n^{-\epsilon}$ fraction of the expectation). Therefore, the result of Theorem 2 is, up to the various constants involved in the statement, the best possible for $O(\log D)$ competitive algorithms.

We give our lower bound on a line network of n vertices, denoted $1, \ldots, n$, with $n = 2^{3s}$, for some s. Observe that on a line network we have $D = n$. Given a randomized $k \log n$-competitive algorithm for the line of n vertices, we present a sequence of requests formed by $l = (1/3) \log n$ phases plus a final extra phase.

At phase i, $i = 0, \ldots.., l$, 2^i pairwise disjoint calls of size $\frac{n}{2^i}$ are presented. These are calls $((j-1)\frac{n}{2^i} + 1, j\frac{n}{2^i} + 1)$, $j = 1, \ldots, 2^i$. Consider now phase number l: During phase $l = (1/3) \log n$, $n^{1/3}$ calls of size $n^{2/3}$ are presented. We will prove that for any randomized $k \log n$-competitive algorithm, at least one of these calls has the property that with probability at least $\frac{1}{6k}$ all the edges included in this call are occupied by the time phase l ends. Then, we present $n^{2/3}$ disjoint calls of length 1 on these edges. The optimal solution is of size $n^{2/3}$, while with probability at least $\frac{1}{6k}$ the on-line algorithm can achieve a benefit of at most $n^{1/3}$, thus showing that the algorithm cannot be

[5] A d-inductive graph is a graph $G = (V, E)$ for which there is an ordering of the vertices such that for any i, $|\{j : i < j, (i, j) \in E\}| \le d$.

close to its expectation with probability $1-o(1)$, even when the size of the optimal solution grows (even with respect to D, and n).

We now turn to prove our claim. A call of phase i is included in one and only one call for each phase $i' < i$, while it is not partially intersected by any other call of phase $i' < i$. On the other hand, a call of phase i includes completely $2^{i'-i}$ calls of a phase $i' > i$.

Let p^i_m be the absolute probability that the m-th call of phase i is accepted (clearly, while all the calls that include it in phases $0, ..., i-1$ are rejected). We denote by P^l_m the sum of the absolute probabilities of acceptance of all the calls that include call m of level l, plus the probability that call m itself is accepted.

We will show that for any $k \log n$ competitive randomized algorithm $\sum_{m=1}^{2^l} P^l_m \geq 2^l \cdot \frac{1}{6k}$. Therefore, there is at least a call m' such that $P^l_{m'} \geq \frac{1}{6k}$, from which the claim follows.

We have:

$$\begin{aligned}
\sum_{m=1}^{2^l} P^l_m &= \sum_{i=0}^{l} 2^{l-i} (\sum_{m=1}^{2^i} p^i_m) \\
&\geq \sum_{i=0}^{l} 2^l \frac{1}{2} \sum_{j=i}^{l} 2^{-j} (\sum_{m=1}^{2^i} p^i_m) \\
&= 2^{l-1} \sum_{j=0}^{l} 2^{-j} \sum_{i=0}^{j} (\sum_{m=1}^{2^i} p^i_m)
\end{aligned}$$

The first equation follows since the absolute probability p^i_m of accepting call m of phase i is summed up for the 2^{l-i} included calls of level l. Now, observe that $\sum_{i=0}^{j} (\sum_{m=1}^{2^i} p^i_m)$ is the algorithm's expected benefit after the calls of phase j have been presented.

We continue with,

$$2^{-j} \sum_{i=0}^{j} (\sum_{m=1}^{2^i} p^i_m) \geq \frac{1}{k \log n} ,$$

since the algorithm is $k \log n$ competitive, and the optimal solution can accept 2^j calls after phase j.

Finally,

$$\sum_{m=1}^{2^l} P^l_m \geq 2^{l-1} \frac{l+1}{k \log n} \geq 2^l \cdot \frac{1}{6k} ,$$

which proves our claim.

4 Further extensions for trees

In this section we show that a better competitive ratio can be obtained, at the expense of a larger deviation. In fact, this algorithm improves upon the best known competitive ratio obtainable for call admission on trees: We achieve a competitive ratio of $4\log 2D$. We also consider extensions of our algorithms to trees with edges of any uniform capacity, and point out that *any* algorithm for call admission on trees can be converted to apply to uniform-capacity trees, with almost the same competitive ratio.

4.1 An algorithm with improved competitive ratio

To obtain the improved competitive ratio we use our general framework, but use a different randomized selection procedure. We refer in the following to the first version of the deterministic filter given in Section 2.2. The randomized selection procedure now uses the "classify and randomly select" technique [ABFR94]. We choose at the beginning at random a number between 1 and $\log 4D$, that we denote A. When a new candidate call is presented to the randomized selection procedure, we first assign the candidate a number between 1 and $\log 4D$, with the only constraint that this number was not assigned to any of the previous candidates that intersect the present one (i.e., we color the candidate calls with a legal coloring in their intersection graph). By Lemma 10, $\log 4D$ numbers are sufficient, since each candidate intersects at most $\log 2D$ previous candidates. The algorithm will now accept those candidates that are assigned the number A. We then obtain the following:

Theorem 12 *There exists a $4\log 4D$-competitive randomized algorithm for call admission on trees of diameter D.*

Proof. We partition $\mathcal{C}$ into a set of $\log 4D$ disjoint classes. Class $\mathcal{C}_i$, $i = 1, \ldots, \log 4D$, contains all the candidates which are assigned number i. All the candidates of a class can be accepted together in a solution since they are non-intersecting. The algorithm accepts a set of calls $\mathcal{A}$ that corresponds to the randomly selected class. The expected size of the on-line solution is $E(\mathcal{A}) \geq \frac{1}{\log 4D}\sum_{i=1}\mathcal{C}_i = \frac{1}{\log 4D}\mathcal{C} \geq \frac{OPT}{4\log 4D}$, since, by Lemma 9, $\mathcal{C} \geq \frac{OPT}{4}$. ■

4.2 Trees with arbitrary uniform capacity

We observe that the proposed algorithms apply also to trees with any uniform capacity for the links. In fact, *any* algorithm for call admission on trees of link capacities 1 can be converted into an algorithm for trees with any uniform integer capacity k, with almost the same competitive ratio.

Given any c-competitive algorithm for the capacity-1 case, we run a "first fit"-based technique [AAFLR96], that uses k copies of the original algorithm on k copies of the tree, but each with capacity 1. If the call is accepted by any of the algorithms we accept it into the actual k-capacity tree. Using [AAFLR96] we get for this case that we have a $c+1$ -competitive algorithm with respect to an adversary that uses k distinct, capacity-1, trees. Since this adversary is only a 1/2 fraction away from the original adversary that has a single tree of uniform capacity k (cf. [AGLR94]), we obtain a $(2(c+1))$-competitive algorithm for the problem.

5 Routing on meshes with high probability

In this section we propose an algorithm for meshes that achieves optimal competitive ratio and a benefit close to its expectation with high probability. We concentrate on those aspects of the algorithm of Kleinberg and Tardos [KT95] (KT in the following) that must be changed to obtain a high probability result, and propose different solutions. KT partitions the $n \times n$ meshes into $\frac{n^2}{\log^2 n}$ squares of size $\log n \times \log n$. Calls that have both endpoints in the same square are *short calls*, calls that have the two endpoints in different squares are *long calls*.

KT first tosses a coin and decides to accept *only* short calls with probability 1/2, or alternatively only long calls. If long calls are accepted, they are restricted to have endpoints only on a specific subset of the squares of the mesh. A crossbar structure is then formed to route long calls through the mesh. This structure must not conflict with short calls and with the routes used by long calls to escape the squares where their endpoints are, and reach the crossbar structure.

We will propose an algorithm that achieves a good fraction of the optimal solution with probability that tends to 1 as the size of the optimal solution grows. We propose a different crossbar structure, that uses ideas from the algorithm of Bartal and Leonardi [BL97] for on-line path coloring on meshes.

The first stage of the algorithm is a *randomized filter* that selects a subset of candidate calls $\mathcal{C}$ of the input sequence σ. Calls of $\mathcal{C}$ form a new input sequence for a complete deterministic procedure. Let us denote with $ON(\mathcal{T})$ and $OPT(\mathcal{T})$, the set of calls of set $\mathcal{T}$ accepted by the on-line and by the optimal solution, respectively, and their cardinality.

We will show that $E(OPT(\mathcal{C}))$ is at least a constant fraction of $OPT(\sigma)$, and that $OPT(\mathcal{C})$ is within a constant fraction of $E(OPT(\mathcal{C}))$ with probability that tends to 1 as the optimal solution grows. For the deterministic procedure, that we denote with ON_D, we also prove $OPT(\mathcal{C}) = O(\log n) ON_D(\mathcal{C})$. The overall algorithm is then $O(\log n)$ competitive, and it is within a constant fraction of its expectation with high probability.

5.1 The randomized filter

We define in any square S of the mesh, three concentric regions: S^1, S^2 and S^3 (see Figure 1). S^1 is the most external region, S^2 is internal to $S^1$and both of them contain $2B = 2\frac{\log n}{13}$ rings of the square. S^3 is internal to both S^1 and S^2, and it is formed by $2B$ concentric rings and by the *central region* of the square of size $B \times B$.

The randomized filter works as follows:

1. Initially, select uniformly at random one of the three regions in each square.
2. Discard call (s, t), if at least one of the two endpoints is in a selected region.

The selected region in each square is part of the crossbar structure, that is especially dedicated to the routing of long calls that cross the square and have endpoints in other squares.

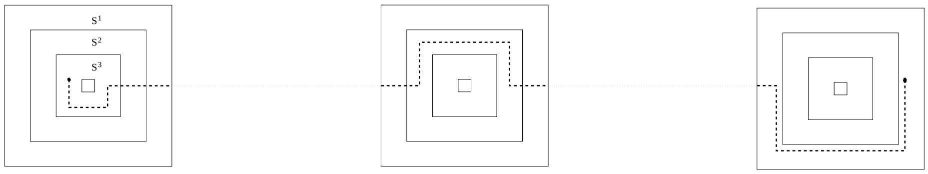

Figure 1: The routing of a long call.

Lemma 13 *For any input sequence σ, $E(OPT(\mathcal{C})) \geq \frac{1}{3}OPT(\sigma)$. For any constant $\delta \in [0,1]$, for any sequence σ with $OPT(\sigma) \geq \frac{72c\log^4 n}{\delta^2}$, $Pr[OPT(\mathcal{C}) \geq (1-\delta)E(OPT(\mathcal{C}))] \geq 1-2exp(-\frac{OPT(\sigma)\delta^2}{72\log^4 n})$.*

Proof. Let us consider the probability that each call of $OPT(\sigma)$ is a candidate call. A short call is a candidate call if both endpoints are outside the selected region. This happens with probability at least 1/3. A long call is a candidate call if both endpoints are outside the selected region of the two squares. This happens at least with probability 4/9. The first part of the claim then follows.

For the second part of the claim we use the "independent bounded differences inequality" in the formulation proposed by Maurey that states the following:

Lemma 14 *[McD89] Let $X_1, \ldots, X_m$ be independent random variables, with X_k taking values in a set A_k for each k. Suppose that the (measurable) function $f : \prod A_k \rightarrow \Re$ satisfies*

$$|f(\underline{x}) - f(\underline{x}')| \leq c_k,$$

whenever the vectors $\underline{x}$ and $\underline{x}'$ differ only in the kth coordinate. Let Y be the random variable $f(X_1, \ldots, X_m)$. Then, for any $t > 0$,

$$Pr[|Y - E(Y)| \geq t] \leq 2exp(\frac{-2t^2}{\sum c_k^2}).$$

In our case we have a set of $m = \frac{n^2}{\log^2 n}$ independent random variables $(X_1, \ldots, X_m)$, associated with the squares of the mesh. Each random variable can assume 1 of 3 values, each with probability 1/3. The function $f(x)$ is in our case $OPT\mathcal{C}(x)$, where $\mathcal{C}(x)$ over all the possible assignments of $x = (X_1, \ldots, X_m)$. A different assignment of a variable X_k, results in a change of the value of $f(x)$ at most equal to the number of short calls and of long calls that have an endpoint in that square and can be accepted in a solution. For this number we have the bound $c_k \leq 4\log^2 n$. In fact, the number of long calls and short calls with at least an endpoint in the square that can be accepted in a solution is at most $4\log n$, the size of the border of the square, and $2\log^2 n$, the number of edges of the square, respectively. For those squares that do not contain calls of the optimal solution, we have $c_k = 0$.

Let K be the number of squares that contain at least one call of the optimal solution. We have $\mu = E(OPT(\mathcal{C})) \geq \frac{1}{3}OPT(\sigma)$, and clearly $OPT(\sigma) \geq K$. ¿From the inequality of Lemma 14 it follows:

$$Pr[OPT(\mathcal{C}) < (1-\delta)\mu] \leq 2exp(\frac{-2(\delta\mu)^2}{\sum_k c_k^2}) \leq 2exp(\frac{-2(\delta\frac{\mathrm{OPT}(\sigma)}{3})^2}{16K\log^4 n}) \leq 2exp(\frac{-OPT(\sigma)\delta^2}{72\log^4 n}),$$

for which the claim is proved.

■

¿From the theorem above, it follows that the probability that the benefit is within a constant fraction of its expectation with probability that tends to 1 as $\frac{OPT(\sigma)}{\log^4 n}$ tends to infinity.

5.2 The deterministic procedure

The deterministic procedure receives as input the calls of $\mathcal{C}$ in the order in which they are presented to the algorithm. We partition $\mathcal{C}$ into the set of candidate long calls $\mathcal{L}$, and the set of candidate short calls $\mathcal{S}$.

Long calls. Long calls have endpoints in two different squares. We first run the following test for each long call (s,t):

- If a call, *short or long*, with one endpoint in the square containing s or t has already been accepted, then discard (s,t).

Let us denote with $\mathcal{L}'$ the subset of calls of $\mathcal{L}$ that pass this test.

We then use the idea of KT, to build a simulated network of higher capacity and run AAP on such network. A vertex in the simulated network is associated with each square of the mesh. Two vertices are connected in the simulated network if the two corresponding squares are adjacent. The bandwidth on such links is equal to $B = \frac{\log n}{13}$, the maximum number of calls that will be routed between two adjacent squares in the on-line solution.

Any call of $\mathcal{L}'$ is transformed into a call between the two vertices of the simulated network associated with the two squares that contain the endpoints of the call, and then submitted to the AAP algorithm.

The call is accepted if it is accepted by the AAP algorithm. The AAP algorithm also indicates a route in the simulated network. This route has an immediate interpretation as a sequence of squares in the original mesh connecting the two squares that contain the endpoints. The call is then routed on a path that is edge disjoint with all previous accepted calls as described in the following. This procedure is similar to that used in [BL97].

We first show how to deal with *external calls*, calls that cross a square but do not have an endpoint there. Since the maximum bandwidth of the on-line algorithm in the simulated network is $B = \frac{\log n}{13}$, it follows that at most B calls are assigned with paths crossing the boundary between two adjacent squares, and there are at most $2B$ external calls for each square.

We will maintain inductively the property that a call crosses the boundary between two squares on a row or on a column connecting the central regions of the two squares. The central region of a square has size $B \times B$. Since B is the maximum number of calls routed between adjacent squares, a distinct row or column can be associated with any call.

By induction, each call enters the square on a row or on a column leading to the central area. We route it towards the central area until a free ring of the selected region is reached. This is always the case since there are at most $2B$ calls crossing the square, and $2B$ available rings in each region. If the selected region is the third one, the $2B$ external rings are still sufficient. As we will show later, short calls and the possible single long call with an endpoint in the square do not cross any edge of the rings in the selected region. The call then follows the ring until it reaches a free row or a free column connecting the central region of the square to the central region of the adjacent square to which the call is directed. The route follows such row or such column until the adjacent square.

Finally, we consider the problem of routing a long call from its endpoint to the border of the square. Observe that no short or long call with an endpoint in this square has been accepted until now. The endpoint of the call is outside the selected region since the call has been candidated. We first consider the case in which the endpoint is outside the central area. The call is routed through the ring on which the endpoint lies. When a free row or column connecting the central area of the square to the central area of the adjacent square to which the call is directed is reached, then it is followed until the appropriate adjacent square. If the endpoint is in the central area, the route follows the ring on which the endpoint is until a "corner". There, it continues on a horizontal or vertical path until any ring among the $2B$ external ones of the third region. There, we continue as for long calls with endpoints outside the central area of the square.

The routing of a long call is shown in Figure 1, where we assume that the second region has been selected in all the three squares.

Short calls. A short call has both endpoints in the same square. We first consider the case when both endpoints are outside the central area or are inside the central area. A short call of this kind is accepted if the two rings have not been assigned to accepted short or long calls with endpoints in the square. The two rings are then completely available, and assigned to the short calls. We bridge the two rings with a horizontal or vertical path that starts from a corner of the internal ring and reaches the external one. These horizontal or vertical paths are edge-disjoint, and do not cross any area connecting the central area of adjacent squares.

We are left with the case in which an endpoint is in the central area and the other is outside the central area. The call is accepted if the two rings on which the two endpoints lie are available. Moreover, if the endpoint outside the central area is in the first or in the second region, a ring between the $2B$ external ones of the third region must also be available. The call follows the ring on which the endpoint in the central area is, until a corner, where a horizontal or vertical path is followed until the ring of the third region that contains the other endpoint, or that has been assigned to the call. In the first case we are done, in the second case a strategy similar to that for calls with both endpoints outside the central area allows to reach the other endpoint of the call.

Figure 2, shows a route of a short call with an endpoint in the central area and one in the first region.

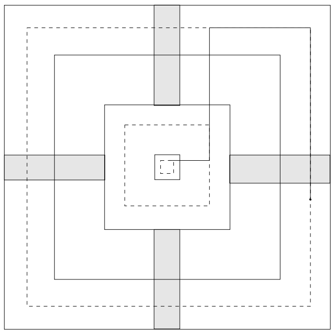

Figure 2: The routing of a short call.

5.3 The analysis of the algorithm

We finally prove that also this algorithm achieves a competitive ratio of $O(\log n)$. By Theorem 13, we know that $E(OPT(\mathcal{C})) \geq 1/3OPT(\sigma)$. We are left to prove that the number of calls accepted by the deterministic procedure is a logarithmic fraction of $OPT(\mathcal{C})$, for any possible set of candidates $\mathcal{C}$.

We first consider long calls. The calls of $\mathcal{L}'$ are submitted to the AAP algorithm on the simulated mesh. Here, the off-line bandwidth is higher that the on-line bandwidth for a constant factor of 13, since at most $\log n$ calls can be routed through 2 adjacent squares. Nevertheless, the algorithm still accepts a number of calls that is an $O(\log n)$ fraction of the optimal solution [AAP93, KT95]. We have $OPT(\mathcal{L}') = O(\log n)ON_D(\mathcal{L}')$, since all the calls accepted by AAP are eventually accepted in the algorithm.

A call (s,t) of $\mathcal{L} - \mathcal{L}'$, is discarded because a short or a long call with an endpoint in the square that contains s or t have already been accepted. However, observe that at most $O(\log n)$ long calls with an endpoint in a square can be accepted in a solution, since the border of the square has logarithmic size. If the algorithm accepts a long call, it can later discard at most a logarithmic number of long calls with endpoint in the two squares containing s and t, that can be accepted in the optimal solution. If the algorithm accepts a short call, it can later discard at most a logarithmic number of long calls with endpoint in the square that contains the two endpoints of the short call. All together, we have $OPT(\mathcal{L} - \mathcal{L}') = O(\log n)(ON_D(\mathcal{S}) + ON_D(\mathcal{L}))$.

We finally consider short calls. We restrict the attention to single square. If we accept a short call, or a long call with an endpoint in the region, we block at most three rings of the square. Then, we can later reject all the short calls that have endpoints on these rings. We should additionally consider that every time a ring of the $2B$ external ones of the third region is assigned, one more call with an endpoint in the central area and the other outside the third region can be refused. Then, $OPT(\mathcal{S}) = O(\log n)(ON_D(\mathcal{S}) + ON_D(\mathcal{L}))$.

It then follows that $OPT(\mathcal{C}) = OPT(\mathcal{L}') + OPT(\mathcal{L} - \mathcal{L}') + OPT(\mathcal{S}) = O(\log n)(ON_D(\mathcal{S}) + ON_D(\mathcal{L})) = O(\log n)ON_D(\mathcal{C})$.

Combining with Lemma 13, we conclude with the following theorem:

Theorem 15 *There exists an* $O(\log n)$*-competitive algorithm for call control on meshes. For any* $\delta \in [0,1]$, *any sequence* σ, $Pr[ON(\sigma) \geq (1-\delta)E(ON(\sigma)] \geq 1 - 2exp(-\frac{OPT(\sigma)\delta^2}{72\log^4 n})$.

Acknowledgment. A special thank is due to Amos Fiat for having drawn our attention to the relationship between the competitive ratio and the deviation of the result from the expectation for randomized on-line algorithms. We also thank Hannah Bast for useful suggestions. The last author would like to thank Demetrios Achlioptas for useful discussions.

References

[AAFPW93] J. Aspens, Y. Azar, A. Fiat, S. Plotkin and O. Waarts. On-line Load Balancing with Applications to Machine Scheduling and Virtual Circuit Routing. *Proc. of STOC'93.*

[AAFLR96] B. Awerbuch, Y. Azar, A. Fiat, S. Leonardi, A. Rosén. On-line Competitive Algorithms for Call Admission in Optical Networks. *Proc. of ESA'96*, LNCS 1136, pp. 431-444.

[AAP93] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput Competitive On-line Routing. *Proc. of FOCS'93.*

[ABFR94] B. Awerbuch, Y. Bartal, A. Fiat, and A. Rosén. Competitive Non-Preemptive Call Control. *Proc. of SODA'94.*

[AGLR94] B. Awerbuch, R. Gawlick, F.T. Leighton, and Y. Rabani. On-line Admission Control and Circuit Routing for High Performance Computing and Communication. *Proc. of FOCS '94.*

[BBKTW90] S. Ben-David, A. Borodin, R.M. Karp, G. Tardos, and A. Wigderson. On the Power of Randomization in Online Algorithms. In *Proc. of the 22nd Ann. ACM Symp. on Theory of Computing*, pp. 379–386, May 1990.

[BFL96] Y. Bartal, A. Fiat, S. Leonardi. Lower Bounds for On-line Graph Problems with Application to On-line Circuit and Optical Routing. *Proc. of STOC'96.*

[BL97] Y. Bartal and S. Leonardi. On-line Routing in All-Optical Networks. To appear in *Proc. of ICALP '97.*

[McD89] C. McDiarmid. On the Method of Bounded Differences. *Surveys in Combinatorics*, J. Siemons ed, Lond. Math. Soc. Lect. Notes Series 141, C.U.P. 1989.

[KR96] J. Kleinberg and R. Rubenfield. Short Paths in Expander Graphs. *Proc. of FOCS '96.*

[KT95] J. Kleinberg and E. Tardos. Disjoint Paths in Densely Embedded Graphs. *Proc. of FOCS'95.*

[LM95] S. Leonardi and A. Marchetti-Spaccamela. On-line Resource Management with Apllications to Routing and Scheduling. *Proc. ICALP '95.*

[L86] M. Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM J. on Computing*, 15 (1986).

[ST85] D. Sleator, R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communications of ACM* 28, 1985.