An improved throughput-competitive algorithm for Online Multicast

Tripurari Singh
Dept. of Computer Science
Johns Hopkins University
tsingh@cs.jhu.edu

December 1999

Abstract

We present an $O(\log^2 n \log \mathcal{M})$ throughout competitive algorithm for online multicast, where n is the number of nodes in the graph, and $\mathcal{M}$ is the number of multicast groups. This improves upon the previous best $O((\log n + \log\log \mathcal{M})(\log n + \log \mathcal{M}) \log n)$ competitive algorithm by Goel, Henzinger and Plotkin. and closes the gap between the upper and lower bounds to $O(\log n)$.

This algorithm uses the recent extension of the maximal dense tree algorithm to graphs with dynamic edge weights.

Our algorithm is modular, and does not involve an integrated analysis of its winner picking and routing components.

1 Introduction

1.1 Problem definition

Multicasting in a communications network is the broadcasting of a signal from its source to a subset of nodes in the network. Point to point connections, or *unicast*, are a special case of multicast.

The throughput competitive online multicast problem is faced by the administrator of a network with limited link capacities. Requests for connections to the various sources arrive online and the network administrator has to make both admission control and routing decisions. That is, he has to decide which requests to accept and how to route connections to accepted requests so that no link is assigned more traffic than its capacity. Furthermore, these decisions must be made online and connection requests that are rejected are lost forever. The network administrator's objective is to maximize the throughput of the network; or more generally the revenue earned by the network, if each requesting node promises money in exchange for connection.

Formally, we are given a capacitated graph $\mathcal{H} = (V, E, U)$, where $u_e \in U$ is the capacity of edge $e \in E$, and a set of $\mathcal{M}$ multicast groups. Each group $g, 1 \le g \le \mathcal{M}$ contains a source vertex s_g that emits a signal of bandwidth σ_g. A sequence of k requests arrive over time, and the network administrator has to make an online admission control and routing decision for each request.

1.2 Previous work

Throughput competitive online routing has a long history. Awerbuch, Azar and Plotkin set the ball rolling with their seminal work [AAP93] in 1993. In this paper they developed the first polylogarithmically competitve algorithm for the *unicast*, or point to point connection problem, and also provided matching lower bounds.

Attention then shifted to multicast problems. The batch multicast problem, wherein all members of a multicast group arrived together as a batch, and an admission control decision for the whole batch had to be made, were studied first. This problem was readily solved using the techniques of [AAP93].

The more general version of the multicast problem allowed each receiver node to independently request connection to a multicast source. The admission control and routing decision was to be made for *each* requesting receiver node. Two further variants of the problem arose: non - interleaved multicast and, the more general, interleaved multicast.

In the non-interleaved multicast variant, *all* requests for connection to one source arrive before requests for the next source. These requests, of course, arrive one at a time, and their admission control and routing decision has to made immediately. The interleaved model allows any node to request connection to any source at any time.

Attempts to solve the non-interleaved variant lead to the definition of the online maximal dense tree problem. Attempts to solve this problem provided the first polylogarithmic approximation to the offline $k - MST$ problem [AABV95], as well an approximation to the *offline* Maximal Dense Tree problem [AAG93].

Finally [AS97] developed a polylogarithmically competitive algorithm for the online Maximal Dense Tree problem. Combining this solution with the edge-cost function of [AAP93] immediately yielded a polylogarithmically throughtput competitive online algorithm for the non-interleaved multicast problem.

Shortly thereafter [GHP98] provided an $O((\log n + \log\log \mathcal{M})(\log n + \log \mathcal{M}) \log n)$ throughut competitive algorithm for the general interleaved multicast problem, where n is the number of nodes, and $\mathcal{M}$ is the number of multicast groups. Their solution required the definition of a new edge-cost function that was independent of the specific admission control decisions made. They also provided a $\Omega(\log n \log \mathcal{M})$ lower bound for the problem; the first bound better than both $\Omega(\log n)$ and $\Omega(\log \mathcal{M})$

Recently, the maximal dense tree algorithm of [AS97] has been generalized in [Sin99], to support dynamic edge weights. In this paper, we use this algorithm to provide an improved competitive online algorithm for the (most general) interleaved multicast problem. We define a deterministic edge cost function that is a slight modification of that used by [AAP93]. Careful analysis of this algorithm yields an $O(\log^2 n \log \mathcal{M})$ bound.

Despite the modifications carried out to the cost function of [AAP93] and the detailed analysis, we hope this paper is simple. It is quite modular, and unlike [GHP98] does not involve an integrated analysis of its winner picking and routing components.

2 Algorithm Description

We denote a request β by a tuple $\beta = (v, g, \rho)$ so that v is understood to be the node requesting a connection to group g, and in return offering revenue ρ. If the connection request β is accepted a path of bandwidth σ_g has to be established from v to a node previously connected to the source s_g, and ρ units of revenue is earned.

The objective of the online algorithm is to maximize the total profit $\tilde{\rho} = \sum_{j=1}^{k} \rho$ without assigning more load (traffic) to any edge e than its capacity u_e. No knowledge of the future - statistical or otherwise - is assumed. Instead the input is assumed to be generated by an oblivious adversary that knows the online algorithm but not the random bits generated by it.

We make the following two assumptions:

Assumption 2.1 $1 \leq \frac{4}{n} \cdot \frac{\rho}{\sigma_g}$, for all requests.

Assumption 2.2 $\sigma_g \leq \frac{\min_{e \in E}(u_e)}{\log \mu}, 1 \leq g \leq \mathcal{M}$

where $\mu = n^2$. These assumptions are similar to those made in [AAP93]. It can be shown that without assumptions similar to these no online algorithm can achieve a polylogarithmic competitive ratio.

We obtain our algorithm by running a distinct Online Maximal Dense Tree algorithm for each group. The Online Maximal Dense Tree algorithm for group g runs on the graph $G_g = (V, E, c(g, j)), 1 \leq g \leq \mathcal{M}, 1 \leq j \leq k$. The graph $G_g = (V, E, c(g, j))$ has the same set of vertices and edges as the graph $\mathcal{H} = (V, E, U)$, but has different edge weights. The weight $c_e(g, j)$ of an edge e, in the graph G_g, at time j, is a function of the bandwidth σ_g of group g, the capacity u_e of the edge e and the traffic load on the edge at time j.

Whenever a connection is accepted by an Online Maximal Dense Tree algorithm it is also accepted by the final online algorithm. The loads of all edges used by this connection are updated and these loads are, in turn, used to re-compute edge weights. We formally present our algorithm, `RouteOrBlock`, in fig 1

Initialize
- $j \leftarrow 1$ /* Initialize request count */
- $\forall e \in E : l_e(1) \leftarrow 0$ /* $l_e(1)$ denotes load on edge e at time 1 */
- $1 \leq i \leq \mathcal{M}, c_e(i, 1) \leftarrow 0$ /* $c_e(i, 1)$ denotes weight of edge e in graph G_i at time 1 */
- $1 \leq i \leq \mathcal{M}$, Initialize an Online Maximal $\frac{1}{2\log(\frac{1}{2\gamma})\cdot(1+\log n)}$-dense Tree
  - with the parameter $\gamma = \frac{1}{4\mathcal{M}}$

RouteOrBlock$(\beta = (g, v, \rho))$
- $\forall e \in E : b_e(j) \leftarrow u_e(\mu^{l_e(j)/u_e} - 1)$
- $\forall e \in E$:**if** $\frac{l_e(j)}{u_e} \leq 1 - \frac{1}{\log \mu}$
  - **then** $c_e(g, j) \leftarrow \frac{\sigma(j)}{u_e} \cdot b_e(j)$
  - **else** $c_e(g, j) \leftarrow \infty$
- Invoke Online Maximal $\frac{1}{2\log(\frac{1}{2\gamma})\cdot(1+\log n)}$-dense Tree algorithm,
  - with $\gamma = \frac{1}{4\mathcal{M}}$, on graph G_g at node v
- $l_e(j+1) \leftarrow l_e(j)$
- **if** request is accepted and fragment $f(j)$ is added by
  - the Online Maximal 1-dense Tree algorithm
  - **then** $\forall e \in f(j), l_e(j+1) \leftarrow l_e(j) + \sigma_g$
  - **else** block the connection
- $j \leftarrow j + 1$

Figure 1: The `RouteOrBlock` algorithm.

It is interesting to note that different Online Maximal Dense Tree algorithms interact with each other only through edge weights, and that each Online Maximal Dense Tree assumes edge weights to be adversarial. This allows the `RouteOrBlock` algorithm to tolerate inter-dependence between its Online Maximal Dense Tree algorithms, such as sharing of random bits. Such sharing of random bits is not possible in the workstation scheduling algorithm of [AAP93].

3 Analysis

Our analysis consists of two parts. We first show that our online algorithm `RouteOrBlock` respects capacity constraints, and then show that it earns revenue that is competitive with the optimal offline algorithm.

3.1 Correctness

Lemma 3.1 `RouteOrBlock` never violates edge-capacity constraints.

Proof: Let $f(j)$ be the the fragment taken as a result of the j^{th} request. Assume to the contrary that fragment $f(j)$ causes edge e to overflow. By assumption 2.2, $\frac{l_e(j)}{u_e} \geq 1 - \frac{1}{\log \mu}$ which implies that the weight of edge e at time $j, c_e(g,j) = \infty$, where g is the group of the j^{th} request. Hence fragment $f(j)$ could not have been formed - a contradiction. ∎

3.2 Competitiveness

We now show that the expected revenue of `RouteOrBlock` is $O(\log^2 n \cdot \log \mathcal{M})$ competitive with the optimal offline on all inputs generated by the oblivious adversary.

Consider group g. Let $\tilde{\tau}_g$ be the tree used by the online algorithm to service g, let $\tilde{w}_g$ be the weight of this tree and let $\tilde{\rho}_g$ be the revenue earned by it. Note that for any given input, $\tilde{w}_g, \tilde{\rho}_g, b_e(j), c_e(g,j)$, are random variables, where $1 \leq j \leq k$ denotes the time. Some of our lemmas will be concerned with the expected values of these random variables, while others will be concerned with their values at specific instances.

Also let $f(j)$ be the fragment taken by `RouteOrBlock` as a result of the j^{th} connection request. Note that $f(j)$ may be null. Let $W(j)$ be the weight of $f(j)$ measured on the cost metric $c_e(g,j)$ of group g at time j, so that $\tilde{w} = \sum_{j=1}^{k} W(j)$. Observe that $W(j)$ is always finite.

Let τ_g^* be the tree used by the optimal offline to service g, and let ρ_g^* be its revenue. Partition τ_g^* into a subtree $\tau_g^{*:dense}$ that is maximally 1-dense in the *final* metric $c(g,k)$, and the remaining sparse forest $\tau_g^{*:sparse} = \tau_g^* - \tau_g^{*:dense}$ attached to it.

Next, partition the set of multicast groups into Q_1 and Q_2. Let Q_1 be the set of all groups g whose online tree contains at least one edge e, of weight ∞, in the final cost metric $c_e(g,k)$. Let Q_2 be the set of remaining groups. In what follows we partition the set of requests served by the offline into 3 sets A, B, C, with revenues $\rho_A^*, \rho_B^*, \rho_C^*$ respectively, so that the total offline revenue $\rho^* = \rho_A^* + \rho_B^* + \rho_C^*$. We then show that the online revenue $\tilde{\rho}$ is competitive with each one of $\rho_A^*, \rho_B^*, \rho_C^*$, thus obtaining our result.

Specifically, let A be the set of connection requests from groups in Q_1 that are accepted by the optimal offline. Let B and C be the sets of requests from groups in Q_2 incident on $\tau_g^{*:dense}$ and $\tau_g^{*:sparse}$ respectively. Note that $\rho_A^*, \rho_B^*, \rho_C^*$ are random variables since the definition of the sets A, B, C depend on the random bits generated by the Online Maximal Dense Tree algorithms. However their sum, $\rho^* = \rho_A^*, \rho_B^*, \rho_C^*$, is not a random variable.

We first bound ρ_C^* in terms of $\tilde{\rho}$.

Lemma 3.2

$$\rho_C^* \leq 2\tilde{\rho}$$

Proof: Consider group g. By definition, $\rho_g^{*:dense}$ is no greater than the revenue of the optimal offline maximal dense tree on $c(g,k)$. Note that unlike $\tau_g^{*:dense}$ this offline maximal dense tree is not restricted to be a subtree of τ_g.

Next we apply lemma 4.1 of [Sin99]. Note that this lemma is applicable even though our graph G_g has infinite weight edges. Since the offline works on the *final* edge weights, it simply ignores the presence of infinite weight edges. The online, on the other hand, takes an edge only if its weight is bounded *at that time*. Since the weight $\tilde{w}_g$ of the online tree is computed using weights of edges at the time they were taken, the online maximal dense tree results of [Sin99] continue to hold. Plugging $d^* = 1, \epsilon = \frac{1}{2}, d = \frac{1}{2\log(\frac{1}{2\gamma})\cdot(1+\log n)}$ and $\gamma = \frac{1}{4\mathcal{M}}$ in lemma 4.1 of [Sin99], we get

$$\rho_g^{*:dense} \leq 2\tilde{\rho}_g \tag{1}$$

where $\tilde{\rho}_g$ is the revenue earned by the online on a group g. Summing over all groups in Q_2 we get,

$$\rho_C^* \leq 2\tilde{\rho} \tag{2}$$

■

Over the next three lemmas we will bound $\rho^*_A + \rho^*_B$ in terms of $\tilde{w}$. This step bears some semblance to the proof of [AAP93]. In the subsequent lemma we will bound $\tilde{w}$ in terms of $\tilde{\rho}$, thus bounding $\rho^*_A + \rho^*_B$ in terms of $\tilde{\rho}$.

In the next lemma we express the revenue ρ^*_g earned by the offline on group $g \in Q_1$ in terms of (a derivative of) the edge weights.

Lemma 3.3 Consider the revenue ρ^*_g of the offline from a group $g \in Q_1$. Then,

$$\rho^*_g \leq \frac{b_e(k)}{u_e} \cdot \sigma_g$$

where e is an edge used by τ^*_g such that $c_e(g,k) = \infty$.

Proof: Recall that $b_e(j)$ is never infinite, $\forall e$ and $\forall j$. Consider an edge e, such that its cost in the final metric $c_e(g,k) = \infty$. By definition,

$$\begin{aligned} \frac{l_e(k)}{u_e} &\geq 1 - \frac{1}{\log \mu} && (3) \\ &= 1 - \frac{1}{2 \log n} && (4) \end{aligned}$$

Therefore,

$$\begin{aligned} b_e(k) &= u_e(\mu^{\frac{l_e(k)}{u_e}} - 1) && (5) \\ &\geq u_e(\mu^{1 - \frac{1}{2\log n}} - 1) && (6) \\ &= u_e(\frac{n^2}{n^{\frac{1}{\log n}}} - 1) && (7) \\ &= u_e(\frac{n^2}{2} - 1) && (8) \end{aligned}$$

Since any non-trivial graph contains at least one node other than the source, $n \geq 2$. Therefore,

$$b_e(k) \geq u_e \cdot \frac{n^2}{4} \qquad (9)$$

Consider the j^{th} request with profit $\rho(j)$ and group g. By assumption 2.1,

$$\rho(j) = \frac{n}{4} \cdot \sigma_g \qquad (10)$$

Since there are at most n connection requests to any group,

$$\rho^*_g \leq \frac{n^2}{4} \cdot \sigma_g \qquad (11)$$

From equations 9, 11 we have,

$$\rho^*_g \leq \frac{b_e(k)}{u_e} \cdot \sigma_g \qquad (12)$$

where e is an edge used by τ^*_g such that $c_e(g,k) = \infty$. ■

In the following lemma we bound $\rho^*_A + \rho^*_B$ in terms of the (derivative of the) edge weights.

Lemma 3.4

$$\rho^*_A + \rho^*_B \leq \sum_{e \in E} b_e(k)$$

where k is the index of the last connection request.

Proof: By definition of $\rho_g^{*:sparse}$,

$$\begin{aligned}
& && (13)\\
\rho_g^{*:sparse} &\leq \sum_{e\in\tau_g^{*:sparse}} c_e(g,k) && (14)\\
&= \sum_{e\in\tau_g^{*:sparse}} \frac{\sigma_g}{u_e} b_e(k) && (15)
\end{aligned}$$

By lemma 3.3 and the above equation we get,

$$\begin{aligned}
\rho_A^* + \rho_B^* &= \sum_{g\in Q_1} \rho_g^* + \sum_{g\in Q_2} \rho_g^* && (16)\\
&\leq \sum_{g\in Q_1}\sum_{e\in\tau_g^*} \frac{\sigma_g}{u_e} b_e(k) + \sum_{g\in Q_2}\sum_{e\in\tau_g^{*:sparse}} \frac{\sigma_g}{u_e} b_e(k) && (17)\\
&\leq \sum_{\forall g}\sum_{e\in\tau_g^*} \frac{\sigma_g}{u_e} b_e(k) && (18)\\
&= \sum_{e\in E} b_e(k) \sum_{\forall g \;\; st \;\; e\in\tau_g^*} \frac{\sigma_g}{u_e} && (19)\\
&\leq \sum_{e\in E} b_e(k) && (20)
\end{aligned}$$

The last inequality follows from the fact that the offline respects capacity constraints. ■

Next, we relate the weight $\tilde{w}$ of the online tree to $\sum_{e\in E} b_e(k)$. This lemma is similar to lemma 3.2 of [AAP93].

Lemma 3.5

$$\tilde{w}\cdot 2\log\mu \geq \sum_{e\in E} b_e(k)$$

where k is the index of the last connection, and $\tilde{w}$ be the weight of the edges bought by `RouteOrBlock`.

Proof: By induction on k. For $k=0$ the inequality is trivially true since both sides are 0. It is enough to show that, if we buy a fragment $f(j)$ at request j of weight $W(j)$.

$$\sum_{e\in E}\{b_e(j+1) - b_e(j)\} \leq 2W(j)\log\mu$$

Consider edge $e\in f(j)$. Using the definition of edge costs, we get:

$$\begin{aligned}
b_e(j+1) - b_e(j) &= u_e(\mu^{\frac{l_e(j)}{u_e}+\frac{\sigma(j)}{u_e}} - \mu^{\frac{l_e(j)}{u_e}}) && (21)\\
&= u_e(\mu^{\frac{l_e(j)}{u_e}}(\mu^{\frac{\sigma(j)}{u_e}} - 1)) && (22)\\
&= u_e(\mu^{\frac{l_e(j)}{u_e}}(2^{log\mu\frac{\sigma(j)}{u_e}} - 1)) && (23)
\end{aligned}$$

By assumption 2.2, we have $\sigma(j)\leq \frac{u_e}{\log\mu}$. Since $2^x - 1 \leq x$ for $0\leq x\leq 1$, we conclude

$$\begin{aligned}
b_e(j+1) - b_e(j) &\leq \mu^{\frac{l_e(j)}{u_e}}\sigma(j)\log\mu && (24)\\
&= (b_e(j)\frac{\sigma(j)}{u_e} + \sigma(j))\log\mu && (25)
\end{aligned}$$

The above upper bound on the change in costs, the fact that the fragment $f(j)$ was formed, and assumption 2.1 imply:

$$\begin{aligned}
\sum_{e\in E}\{b_e(j+1)-b_e(j)\} &\leq \log\mu \sum_{e\in f(j)} (b_e(j)\frac{\sigma(j)}{u_e}+\sigma(j)) && (26)\\
&\leq \log\mu(W(j)+|\{e\in f(j)\}|\cdot\sigma(j)) && (27)\\
&\leq 2W(j)\log\mu && (28)
\end{aligned}$$

■

We now relate $E(\tilde{w}), E(\tilde{\rho})$ and the density d of the Online Maximal Dense Trees. Given lemma 4.2 of [Sin99] this task would have been simple, but for the error term $\gamma R_g, 1\leq g\leq \mathcal{M}$. In the following lemma we efficiently absorb the error term γR_g and show that $d\cdot E(\tilde{w})\leq 2E(\tilde{\rho})$. Note that we do not show $d\cdot E(\tilde{w}_g)\leq 2E(\tilde{\rho}_g)$ for every individual group.

Lemma 3.6 Let $E(\tilde{w})$ be the sum of the weights of edges taken by `RouteOrBlock` and let $E(\tilde{\rho})$ be the revenue earned by it. Then,

$$d\cdot E(\tilde{w})\leq 2E(\tilde{\rho})$$

Proof: From lemma 4.2 of [Sin99] we have,

$$d\cdot E(\tilde{w}_g) \leq E(\tilde{\rho}_g)+\gamma R_g \qquad (29)$$

where R_g is the revenue used by the Online Maximal Dense Tree of group g on its first Greedy Resurrection.

It is important to note that the expectations in the above inequality are *only* with respect to the random bits used by the the online maximal dense tree algorithm for group g. The random bits used by other groups are reflected in the edge weights and are treated as input by the Online Maximal Dense Tree algorithm for group g. $E(\tilde{w}_g), E(\tilde{\rho}_g)$ and R_g, as used in the above inequality, are random variables in the context of `RouteOrBlock`.

In what follows, we assume the context of `RouteOrBlock` and take expectation over random bits generated by *all* groups. From equation 29 we have,

$$d\cdot E(\tilde{w}_g) \leq E(\tilde{\rho}_g)+\gamma E(R_g) \qquad (30)$$

To see this, observe that the random bits generated by groups other than g are treated as input by the Online Maximal Dense Tree algorithm for g and that the inequality 29 holds for *all* inputs.

Summing over all groups we have,

$$\begin{aligned}
d\cdot E(\tilde{w}) &\leq d\cdot\sum_{g=1}^{\mathcal{M}} E(\tilde{w}_g) && (31)\\
&\leq \sum_{g=1}^{\mathcal{M}} E(\tilde{\rho}_g)+\gamma\sum_{g=1}^{\mathcal{M}} E(R_g) && (32)
\end{aligned}$$

Next we express $E(R_g)$ in terms of $E(\tilde{w})$. By definition of R_g,

$$\begin{aligned}
R_g &\leq d\sum_{e\in\tau} c_e(g,k) && (33)\\
&\leq d\sum_{e\in E} c_e(g,k) && (34)\\
&= d\sum_{e\in E}\frac{\sigma_g}{u_e}b_e(k) && (35)
\end{aligned}$$

By assumption 2.2, $\frac{\sigma_g}{u_e} \leq \frac{1}{\log \mu}, \forall g$. Therefore,

$$R_g \leq d \cdot \frac{1}{\log \mu} \sum_{e \in E} b_e(k) \tag{36}$$

By lemma 3.5

$$R_g \leq d \cdot 2\tilde{w} \tag{37}$$

Note that both R_g and $\tilde{w}$ are random variables. Taking expectations we get,

$$E(R_g) \leq d \cdot 2E(\tilde{w}) \tag{38}$$

Substituting in inequality 32 we get,

$$d \cdot E(\tilde{w}) \leq E(\tilde{\rho}_g) + \mathcal{M}\gamma \cdot 2d \cdot E(\tilde{w}) \tag{39}$$

Since $\gamma = \frac{1}{4\mathcal{M}}$ we have the result,

$$d \cdot E(\tilde{w}) \leq 2E(\tilde{\rho}_g) \tag{40}$$

■

We now connect the previous lemmas to obtain our final result. Recall that our basic argument is to bound each of $\rho^*_A, \rho^*_B, \rho^*_C$ in terms of the expected online revenue $E(\tilde{\rho})$. Specifically, lemma 3.2 handles ρ^*_C, while lemmas 3.4, 3.5 and 3.6 handle ρ^*_A and ρ^*_B.

Theorem 3.7 The online `RouteOrBlock` algorithm does not violate capacity constraints and its expected revenue is $O(\log^2 n \cdot \log \mathcal{M})$ competitive with the optimal offline.

Proof: From lemmas 3.4, 3.5 we have,

$$\rho^*_A + \rho^*_B \leq \sum_{e \in E} b_e(k) \tag{41}$$

$$\leq \tilde{w} \cdot \log \mu \tag{42}$$

Taking expectations,

$$E(\rho^*_A) + E(\rho^*_B) \leq E(\tilde{w}) \cdot \log \mu \tag{43}$$

Plugging in $E(\tilde{w})$ and $d = \frac{1}{2\log(2\mathcal{M}) \cdot (1+\log n)}$ in lemma 3.6 we have,

$$E(\rho^*_A) + E(\rho^*_B) \leq E(\tilde{\rho}) \cdot 4\log(2\mathcal{M}) \cdot (1 + \log n) \log \mu \tag{44}$$

Noting that lemma 3.2 holds for every sequence of random bits generated, we have,

$$E(\rho^*_C) \leq 2E(\tilde{\rho}) \tag{45}$$

Recall that ρ^* is not a random variable. Now,

$$\rho^* = E(\rho^*) \tag{46}$$

$$= E(\rho^*_A) + E(\rho^*_B) + E(\rho^*_C) \tag{47}$$

$$\leq E(\tilde{\rho}) \cdot 4\log(2\mathcal{M}) \cdot (1 + \log n) \cdot \log \mu + 2E(\tilde{\rho}) \tag{48}$$

$$\leq E(\tilde{\rho}) \cdot (4\log(2\mathcal{M}) \cdot (1 + \log n) \cdot \log \mu + 2) \tag{49}$$

Since $\mu = n^2$ we conclude that the expected revenue of `RouteOrBlock` is $O(\log^2 n \cdot \log \mathcal{M})$ competitive with that of the optimal offline algorithm. ■

4 Future Work

Our algorithm closes the gap between the upper and lower bounds of the problem to within $O(\log n)$. We conjecture that signficant gains can be achieved in raising the lower bound. The current lower bound, by [GHP98], is a combination of the lower bounds for unicast routing [AAP93] and winner picking [AAFL96]. It might be possible to also factor in the lower bound for the online steiner tree problem [IW91].

References

[AABV95] Baruch Awerbuch, Yossi Azar, Avrim Blum, and Santosh Vempala. Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. In *Proceedings of the ACM STOC*, May 1995.

[AAFL96] Baruch Awerbuch, Yossi Azar, Amos Fiat, and Tom Leighton. Making commitments in the face of uncertainty: How to pick a winner almost every time. In *Proceedings of the ACM STOC*, 1996.

[AAG93] Baruch Awerbuch, Yossi Azar, and Rainer Gawlick. Dense trees and competitive selective multicast. unpublished manuscript, December 1993.

[AAP93] Baruch Awerbuch, Yossi Azar, and Serge Plotkin. Throughput competitive on-line routing. In 34^{th} *Annual Symposium on Foundations of Computer Science,* Palo Alto, California, pages 32–40. IEEE, November 1993.

[AS97] Baruch Awerbuch and Tripurari Singh. Online algorithms for selective multicast and maximal dense trees. In *Proceedings of the* 29^{th} *Annual ACM Symposium on Theory of Computing, El Paso, Texas*, May 1997.

[GHP98] Ashish Goel, Monika R. Henzinger, and Serge Plotkin. An online throughput-competitve algorithm for multicast routing and admission control. 1998.

[IW91] M. Imase and B.M. Waxman. Dynamic steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, august 1991.

[Sin99] Tripurari Singh. The maximal dense tree problem on a graph with dynamic edge weights. Technical Report CNDS-99-4, The Johns Hopkins University, 1999.