Top-down algorithms for loss-based multicast tree construction

Suprakash Datta

Technical Report CSE-2006-08

September 1, 2006

Department of Computer Science and Engineering

4700 Keele Street North York, Ontario M3J 1P3 Canada

TOP-DOWN ALGORITHMS FOR LOSS-BASED MULTICAST TREE CONSTRUCTION

Suprakash Datta
Computer Science and Engineering Department,
York University, Toronto, Canada

ABSTRACT
While robust measurements of network dynamics are essential for the design and management of internetworks, administrative, economic and privacy issues make it impractical to monitor every link in the network. Therefore it becomes necessary to infer network and performance characteristics from end-to-end measurements. One characteristic that is frequently needed to build reliable multicast protocols is the topology of the tree used for multicast communications. Several algorithms have been proposed in the literature for making use of correlations between packets lost at each of the receivers to infer a multicast tree. Most algorithms in the literature are *centralized* – they require some node (e.g., the source) to gather data from all receivers in order to compute the tree. Also, most algorithms build the tree up in a bottom-up manner, so that the entire tree must be built even if we require few levels from the top. In this work, we propose efficient top-down algorithms for the multicast tree inference problem. We show that our algorithm can be easily implemented in a distributed fashion, which improves the scalability. We analyze our algorithms and prove upper bounds on the *sample complexity* or the number of samples required to infer multicast trees with given error bounds and sensitivity. We prove that these bounds are tight by proving matching lower bounds on the sample complexity for some trees. Our results quantify the tradeoff between the number of samples and the probability of computing the tree correctly, and provide lower bounds on the number of samples needed to compute the tree with a given error probability.

1 Introduction

Several modern applications, including software updates, anti-virus patches, shared whiteboards, video conferences and streaming multimedia applications are built using IP multicast. They function by sending packets along a tree rooted at the sender. All packets to a receiver follow the unique path along the tree from the sender to it. Multicast trees are set up in a distributed manner as receivers join the session, and the source of the multicast has no direct knowledge of the tree that is used to perform the multicast. However, many applications require reliable transfer of data, and since IP does not provide this reliability, it must be built in the application layer. As pointed out in [17], the choice of the loss recovery algorithm in the design of a reliable multicast algorithm has a huge effect on its performance, especially scalability. Global loss recovery algorithms retransmit packets lost at any receiver(s) to all receivers and are thus not scalable. Thus virtually all efficient reliable multicast algorithms (e.g. SRM [10], SOT [13] and RMTP [14]) use local recovery – i.e., packet retransmissions are done locally, in the vicinity of the receiver which did not receive the packet. Not surprisingly, much of the multicast tree needs to be computed in order to determine the nodes in the vicinity of a given node. However, this information cannot be obtained by querying routers in real networks, and has to be inferred from end-to-end measurements[1].

In this paper, we study the problem of computing the multicast tree from end-to-end measurements of packet losses. Specifically, we assume that a number of *probe* packets are multicast and the receivers record the list of probe packets they receive. This information is used (e.g., by the source) to infer the multicast tree – i.e., both the topology of the logical multicast tree as well as the edge weights (in our model the weight of an edge is the probability that a packet does not get dropped while traversing the edge – our model will be defined precisely in Section 2). We note that since we use loss measurements to estimate the tree, it follows that we can only hope to compute the *logical* tree. The logical tree differs from the physical tree in that each edge in the logical tree could correspond to a path in the physical tree.

1.1 Previous work

We make no attempt to provide an exhaustive survey of the literature, and refer the interested reader to a survey [6] and to the webpage of the MINC project (`http://www-net.cs.umass.edu/minc/`). In this subsection, we mention papers that are relevant to this work.

In [17] the authors propose an algorithm for the bottom-up construction of the multicast tree.They do not provide any performance analysis, and the algorithm uses parameters which need to be hand-tuned to get good performance.

In [3] the problem is defined rigorously and the authors provide algorithms for bottom-up tree construction. They also prove that these algorithms converge asymptotically to the

[1]This area has received a lot of attention in recent years as a tool for network tomography; see [6] for a survey.

unique logical tree corresponding to the real tree. Further they show that these algorithms find the maximum likelihood estimator trees. Our work is complementary to this work for the following reason. Maximum likelihood estimators provide "optimal" *estimates* since they compute the structure that has the maximum likelihood (among all possible structures) of having produced the set of observations used to generate the estimate. However the use of maximum likelihood estimation provides no guarantees about the quality of the solution obtained, and it is difficult to compute in closed-form how many samples we need to generate a solution of a given quality. In particular, computing an expression for the confidence interval and using the expression to compute the number of samples does not yield closed-form expressions for the sample size.

In [9], the authors study the problem of network inference in the presence of missing data. A recent paper [16] studies multiple source multiple destination tomography.

We observe that all the above papers, and indeed most papers in the literature, propose bottom-up algorithms (also called agglomerative algorithms) in which a forest is maintained and trees in the forest merged until a single tree is produced. However, for many applications, top-down algorithms may be more useful. E.g., for most local loss recovery algorithms for multicast applications, the top levels of the tree are more useful than the bottom levels; in fact the whole tree need not be known. Another advantage of a top-down algorithm is that the upper levels of the tree are known before the algorithm terminates; this information can be obtained using bottom-up algorithms only after the algorithm has terminated.

Finally, we mention for the sake of completeness that several authors (e.g, Rubenstein et al [18, 7]) use unicast-based techniques to estimate network characteristics. We do not survey this area as it is outside the scope of our work.

1.2 Our contributions

The main contributions of this paper are as follows.

Top-down tree construction: We propose a simple top-down algorithm for constructing a multicast tree and show that its sample complexity is optimal within constant factors. We also indicate a simple scheme to prevent the propagation of errors in computing the tree.

Distributed implementation: Unlike most algorithms, our algorithm can be implemented in a distributed manner, so that data from all the receivers do not have to be collected at a single node.

Analysis of the sample complexity: Since the multicast tree must be estimated from a finite number of measurements, it follows that any algorithm outputs an incorrect result with non-zero probability. Intuitively, one expects that the *reliability* of estimation of the multicast tree to increase with the number of samples. However, this tradeoff between the reliability and the number of samples has not been fully characterized. We study the precise dependence of the reliability on the *sample complexity* (i.e. the number of probe packets) and the sensitivity (ability to distinguish edges with weights 1 and $1-\epsilon$), and prove matching upper and lower bounds on the sample complexity of our algorithms. This bounds are closed-form functions of the different parameters of the problem, and quantify the tradeoff between the reliability of a tree-estimation algorithm and the sample complexity. An interesting consequence of our analyses is that it confirms the intuitive idea that the complexity estimating the tree topology alone is less than the complexity of estimating the topology as well as the weight of each edge of the tree.

2 Our model and problem description

The network model As mentioned before, we assume that the multicast tree has to be determined from end-to-end measurements, and that the tree nodes are not required to furnish any information to the algorithm. To simplify the exposition, we assume that the receivers are at the leaves of the logical tree. We define a logical multicast tree to be a

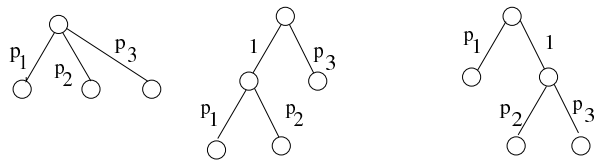

Figure 1. Equivalent multicast trees

tree with arbitrary degrees which has a weight $p(i,j)$ associated with every edge (i,j) of the tree. We define the weight of the path from the root to any leaf as the product of the weights of the edges in the path. Note that a tree with arbitrary node degrees can be represented as an *equivalent* binary tree which has the same set of receivers and the weight of the paths to each receiver has the same weight as the original tree (see Figure 1 for an example). The equivalent binary tree is constructed by introducing some edges with weight 1, and has at most twice the number of nodes and twice the number of edges as the original tree.

The packet loss model We assume that a packet traversing link (i,j) at timestep t is lost with probability $\ell(i,j)$ (called the *loss probability*), and this decision is independent of all other random decisions made by the network. Let us denote as *success probability* the probability that a packet is not dropped, i.e., $p(i,j) = 1-\ell(i,j)$. We use the success probability $p(i,j)$ as the weight of edge $(i.j)$. We assume that for all edges (i,j) in the logical multicast tree that our algorithm outputs, $1-\epsilon > p(i,j) > \epsilon$.

This assumption is clearly an approximate model of real multicast networks, since we know there are temporal and spatial correlations in networks. This assumption is usually made in the literature [17, 3], and is a good approximation

of reality in the case where loss rates are low. Also, our approach is not well-suited to networks with very low packet loss probabilities. In such networks delay-based topology inference methods (see, e.g., [7]) may be more useful.

2.1 Tree estimation vs. edge weight estimation

Intuitively, the task of estimating the topology of a multicast tree should be easier than estimating the topology as well as the edge costs. We would like prove this in this paper. Hence we distinguish between these two estimation problems in this work.

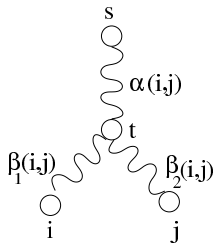

Figure 2. Path weights (success probabilities along paths)

The labels α, β_1, β_2 shown in Figure 2 represent the weights (success probabilities) on the *paths* $(s,t),(t,i),(t,j)$ respectively. The tree estimation problem is to find the correct logical multicast tree and the edge weight estimation problem is to find the logical tree and the weights on the edges.

2.2 Reliability of topology inference

In this section we define two parameters δ and ϵ which quantify the reliability of any algorithm for our problem. These parameters are named after the very similar parameters used in PAC learning in computational learning theory [12] (we will define ϵ slightly differently to suit our needs). The parameter δ is the probability that the algorithm *fails* – i.e., it makes an error in determining the multicast tree. The parameter ϵ is the *sensitivity* parameter for the problem. We will require our algorithm to distinguish two edges only if their weights differ by at least ϵ. We assume that for all edges (i,j) in the logical multicast tree the edge weight $p(i,j)$ satisfies $\epsilon \leq p(i,j) \leq 1-\epsilon$. Figure 4 shows two trees with different topologies in which all paths from the root to leaves have the same weights. The value of $\alpha(a,b)$ differs by ϵ in the two topologies. Thus any algorithm that must distinguish between the trees must be able to to distinguish between the $\alpha()$ values 1 and $1-\epsilon$. The parameter ϵ is an input to any algorithm for this problem, since one expects that the sample complexity of the algorithm would increase as ϵ decreases, even if δ does not change.

3 Topology inference of multicast trees

Several existing tree inference algorithms (e.g., [17]) compute multicast trees from the weights α, β_1, β_2 (shown in Figure 2) for every pair of receivers. These can be estimated from the list of probe packets received (henceforth called *transcripts*) at the receivers i, j in the following way. Let S_{ij} be the *fraction* of probes that reach (i) and (j), $S_{i\bar{j}}$ the fraction that reaches (i) but not (j), $S_{\bar{i}j}$ the fraction that reaches (j) but not (i), and $S_{\bar{i}\bar{j}}$ the fraction that reaches neither (i) nor (j). Then the probabilities α, β_1, β_2 can be estimated using the following estimates. It can be shown easily that the estimates converge to the probabilities in the limit; the details are omitted.

$$\alpha(i,j) \approx \frac{(S_{ij}+S_{\bar{i}j})(S_{ij}+S_{i\bar{j}})}{S_{ij}} \quad (1)$$

$$\beta_1(i,j) \approx \frac{S_{ij}}{S_{ij}+S_{\bar{i}j}} \quad (2)$$

$$\beta_2(i,j) \approx \frac{S_{ij}}{S_{ij}+S_{i\bar{j}}} \quad (3)$$

The algorithm we present next also uses these quantities.

3.1 A centralized top-down algorithm for multicast tree inference

The basic idea behind our top-down algorithm TD is very simple and described below. It is somewhat similar to hierarchical clustering (see, e.g.,[4]). For simplicity of exposition we assume that every non-leaf node has two children. The algorithm can be easily modified to remove this restriction, but we omit the details. Define $f \cong g$ if $|f-g| \leq \epsilon/2$. Let S be the set of receivers.

1. Create a root node (corresponding to the sender s). Let $U \leftarrow S$.
2. Find a pair of nodes x, y that have the highest $\alpha(x,y) = \alpha_{\max}(U)$ and partition the set of receivers U into two sets S_x, S_y such that for any $i \in S_x, j \in S_y$, $\alpha(i,j) \cong \alpha_{\max}(U)$. Create tree nodes h, k corresponding to S_x, S_y. The weights of the edges (s,h) and (s,k) will be set after the next step.
3. Apply step 2 recursively on S_x, S_y, i.e., set $U \leftarrow S_x$ and repeat step 2. Set the weight of edge (s,h) to be $\alpha_{\max}(S)/\alpha_{\max}(S_x)$.

   Then set $U \leftarrow S_y$ and repeat step 2. Set the weight of edge (s,k) to be $\alpha_{\max}(S)/\alpha_{\max}(S_y)$.

The algorithm above recursively partitions sets into two based on the α, β_1, β_2 values to produce a binary multicast tree. There are two important issues that need to be addressed. First, multicast trees need not be binary. This is easily handled by traversing the binary tree and collapsing edges with weight $w \cong 1$. Second, we want an algorithm with an optimal sample complexity. The above algorithm

accumulates errors each time the $\alpha()$ values are updated, and this means that the effective ϵ value decreases quickly, which in turn necessitates a larger number of probes to maintain the same value of δ. We solve this problem using the following idea. Instead of computing the α, β_1, β_2 values for pairs of nodes, we compute similar parameters $\phi(i,j,k)$ for triples of nodes $\langle i,j,k \rangle$. Interestingly, these parameters can be computed efficiently computed from the the α, β_1, β_2 values computed for pairs of nodes. By using the $\phi(i,j,k)$ values, we can prevent errors from accumulating at each level. We omit the details due to space constraints.

3.2 Distributed implementation

Any centralized algorithm suffers from scalability problems – if a single node (e.g., the source s) has to collect all n receiver transcripts of size T each, the network around the source gets congested when n is large. Thus it is desirable to compute the multicast tree in a manner that does not require all transcripts to be gathered at a single node.
We describe now a distributed implementation of algorithm TD. In this implementation, the traffic going to the source is of size $\Theta(n+T)$ instead of $\Theta(nT)$ as in the case of the centralized algorithm. The key idea is to distribute the tree computation steps to the receivers. Note that this also allows parts of the tree to be constructed in parallel.

1. Source s chooses distinct receivers a, b uniformly at random, and gets their transcripts. It multicasts those transcripts and the addresses of a, b to all receivers.

2. Each receiver r computes α, β_1, β_2 parameters for the pairs $(r,a),(r,b),(a,b)$.

3. If $\alpha(r,a), \alpha(r,b) > \alpha(a,b)$, then WLOG assume $\alpha(r,a) \geq \alpha(r,b)$. Then r sends s the value of $\alpha(r,a)$. Else $\alpha(r,b) < \alpha(a,b)$ or $\alpha(r,a) < \alpha(a,b)$. WLOG assume $\alpha(r,a) \leq \alpha(r,b)$. Then r sends a message to s saying that it is in a's subtree S_a.

4. If the source s only gets messages from all receivers saying that they are in subtrees of a or b, then s multicasts to all receivers that they should contact the node (i.e., a or b) whose subtree they are in. Nodes a and b now run the same algorithm as s but only on their respective subtrees S_a, S_b.

   If the source gets a message from at least one node r containing the value $\alpha(r,a) > \alpha(a,b)$ then it chooses the r' with the largest $\alpha(r',a)$ value and sets $b = r'$, and goes back to step 1.

Note that the last step ensures that a and b are chosen so that they lie in subtrees rooted at different children of the root. In the worst case, several rounds are needed before the desired a, b are found. Once such a, b are found, we use a divide-and-conquer approach.
We note that the root node does communicate with all receivers but the amount of information received from each node is very small, since it does not receive transcripts from every receiver. Further, this coordination seems to be necessary since no node (including the source) knows all the receivers *a priori*.

4 Experiments

We have simulated the model used in this paper using a homegrown C program, and also implemented the algorithm in the ns-2 network simulator [1]. This research is work in progress, and we have not finished our experimental evaluation of the algorithms, and the results presented here do not represent a comprehensive study.
We simulated using ns, two 4-leaf binary trees, one balanced and one unbalanced. We ran 100 trials with different random numbers and used them to compute the success probability $(1-\delta)$of the algorithm. We plotted the number of probe packets against $\log(\frac{1}{\delta})$. The observed curve is very nearly linear. This relationship was confirmed by our analytical results presented later.

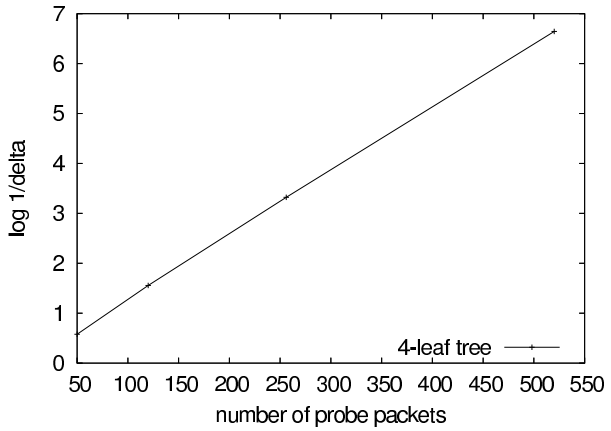

Figure 3. Variation of $1/\delta$ with m

We also simulated complete binary trees with 8 and 16 leaves. We computed the number of probe packets required to get success probability of 0.95. The graph of the number of probe packets was also seen to be linear with $\log n$. We omitted this graph to make space for the analysis in the next section.

5 Analytical Results

We now state our analytical results for our multicast tree inference algorithms. The first result is a lower bound on the sample complexity for estimating the weight of a single edge.

5.1 Lower bounds for a single edge

For technical reasons, we will assume $\delta < 0.07\epsilon$. The bounds are in terms of a rate function $\mathbb{I}(x)$ that has been

used in the theory of large deviations [8] that has been used to study different problems in communication networks [19]. $\mathbb{I}(x)$ is defined as follows.

$$\mathbb{I}(x) = x \ln \frac{x}{p} + (1-x) \ln \frac{1-x}{1-p} \tag{4}$$

Lemma 1 *The number of probes, m, required by any algorithm to distinguish between two edges with weights p and $p+\epsilon$ with failure probability at most $\delta > 0$ satisfies*

$$m(\epsilon, \delta) \geq \frac{\ln[2\mathbb{I}\left(p+\frac{\epsilon}{2}\right)\frac{c^2}{\delta^2}]}{4\mathbb{I}\left(p+\frac{\epsilon}{2}\right)}$$

Proof: We assume that the *a priori* probabilities of the two edges are equal. We classify an edge by comparing the observed probability (fraction) of received probes with the threshold $p+\epsilon/2$. Thus, an edge is classified correctly when its weight is estimated to within $\epsilon/2$ of its correct value. We need the probability of misclassifying the edge to be no more than δ. Since the *a priori* probabilities are equal, each edge must be classified with this error probability. Without loss of generality, assume that the edge being classified has weight p. In order to find a lower bound on the number of probes required to ensure an error probability of at most δ, we can set the lower bound on the probability $\text{Prob}(S/n > p+\epsilon/2)$ to be at most δ. From Theorem 12 in the Appendix, if $n > \frac{1}{x-p}$, then

$$\begin{aligned} \text{Prob}(S/n > p+\epsilon/2) &= \text{Prob}(S > (p+\epsilon/2)n) \\ &\geq \frac{c}{\sqrt{n}} \exp(-n\mathbb{I}(p+\epsilon/2)). \end{aligned}$$

Therefore,

$$\frac{c}{\sqrt{n}} \exp\left(-n\mathbb{I}\left(p+\frac{\epsilon}{2}\right)\right) \leq \delta, \tag{5}$$

where $c = \frac{0.15p(1-x)}{\sqrt{x}(1-p)^{3/2}} = \frac{0.15p(1-p-\epsilon/2)}{\sqrt{(p+\frac{\epsilon}{2})}(1-p)^{3/2}}$ for $x = p+\frac{\epsilon}{2}$. Thus, we have

$$\begin{aligned} \frac{c^2}{n} \exp\left(-2n\mathbb{I}\left(p+\frac{\epsilon}{2}\right)\right) &\leq \delta^2 \\ n \exp\left(2n\mathbb{I}\left(p+\frac{\epsilon}{2}\right)\right) &\geq \frac{c^2}{\delta^2} \\ 2n\mathbb{I}\left(p+\frac{\epsilon}{2}\right) \exp\left(2n\mathbb{I}\left(p+\frac{\epsilon}{2}\right)\right) &\geq 2\mathbb{I}\left(p+\frac{\epsilon}{2}\right)\frac{c^2}{\delta^2} \end{aligned}$$

Using Lemma 14 and Lemma 15, it follows that

$$2n\mathbb{I}\left(p+\frac{\epsilon}{2}\right) \geq \ln\left(\frac{A}{\ln A}\right)$$

where $A = 2\mathbb{I}\left(p+\frac{\epsilon}{2}\right)\frac{c^2}{\delta^2}$. So,

$$\begin{aligned} n &\geq \frac{\ln\left(\frac{A}{\ln A}\right)}{2\mathbb{I}\left(p+\frac{\epsilon}{2}\right)} \\ &\geq \frac{\ln A}{4\mathbb{I}\left(p+\frac{\epsilon}{2}\right)} \text{ by Lemma 13} \\ &= \frac{\ln[2\mathbb{I}\left(p+\frac{\epsilon}{2}\right)\frac{c^2}{\delta^2}]}{4\mathbb{I}\left(p+\frac{\epsilon}{2}\right)} \end{aligned}$$

■

Corollary 2 *When $p = 1-\epsilon$, the number of probes, m, required by any algorithm to distinguish between two edges with weights p and $p+\epsilon$ with failure probability at most $\delta > 0$ satisfies*

$$m(\epsilon, \delta) = \Omega\left(\frac{1}{\epsilon}\left(\ln\frac{1}{\delta}\right)\right)$$

Proof: Substituting $p = 1-\epsilon$ in Lemma 1, we have

$$\begin{aligned} m &\geq \frac{\ln[2\mathbb{I}\left(1-\frac{\epsilon}{2}\right)\frac{c^2}{\delta^2}]}{4\mathbb{I}\left(1-\frac{\epsilon}{2}\right)} \\ &= \frac{\ln[2\mathbb{I}\left(1-\frac{\epsilon}{2}\right)]}{4\mathbb{I}\left(1-\frac{\epsilon}{2}\right)} + \frac{\ln[\frac{c^2}{\delta^2}]}{4\mathbb{I}\left(1-\frac{\epsilon}{2}\right)} \\ &> \frac{\ln[\frac{c^2}{\delta^2}]}{4\mathbb{I}\left(1-\frac{\epsilon}{2}\right)} \\ &\geq \frac{(1-\epsilon)\ln[\frac{c^2}{\delta^2}]}{\epsilon} \\ &\geq \frac{1}{2\epsilon}\ln\left[\frac{0.0225(1-\epsilon)^2\epsilon}{2\epsilon^3(1-0.5\epsilon)\delta^2}\right] \\ &= \Omega\left(\frac{1}{\epsilon}\left(\ln\frac{1}{\delta}\right)\right) \end{aligned}$$

■

It can be shown that the lower bound $m(\epsilon, \delta)$ on the number of probes is maximized when $p = 0.5$.

Corollary 3 *When $p = \frac{1}{2}$, the number of probes, m, required by any algorithm to distinguish between two edges with weights p and $p+\epsilon$ with failure probability at most $\delta > 0$ satisfies*

$$m(\epsilon, \delta) = \Omega\left(\frac{1}{\epsilon^2}\left(\ln\frac{1}{\delta}\right)\right)$$

Proof: Substituting $p = \frac{1}{2}$ in Lemma 1, we have $m \geq \frac{\ln[2\mathbb{I}\left(\frac{1+\epsilon}{2}\right)\frac{c^2}{\delta^2}]}{4\mathbb{I}\left(\frac{1+\epsilon}{2}\right)}$. From Lemma 17, $\mathbb{I}\left(\frac{1+\epsilon}{2}\right) < \epsilon^2$. Therefore, $m \geq \frac{1}{\epsilon^2}\ln\frac{c^2}{\delta^2}$. For $p = \frac{1}{2}$, $c^2 = \frac{0.045(1-\epsilon)^2}{1+\epsilon} = \Theta(1)$, since $c^2 \leq 0.045$ and $c^2 \geq 0.0075$, using $0 < \epsilon < 0.5$. ■

NOTE: It is worth pointing out that this simple algorithm is exactly the Bayesian classifier for the problem of classifying end-to-end observations to one of the two trees (i) and (ii), given that their *a priori* probabilities are equal. It is known that the Bayesian classifier is optimal in the sense that it minimizes the probability of error (see, e.g., [11]).

5.2 Lower bounds for general trees

Consider the topologies in Figure 4 (i) and (ii). A distinguishing feature between the two trees that is observable from end-to-end measurements is the shared loss rate between nodes (a) and (b), i.e., the $\alpha(a,b)$ values are $1-\epsilon$ and

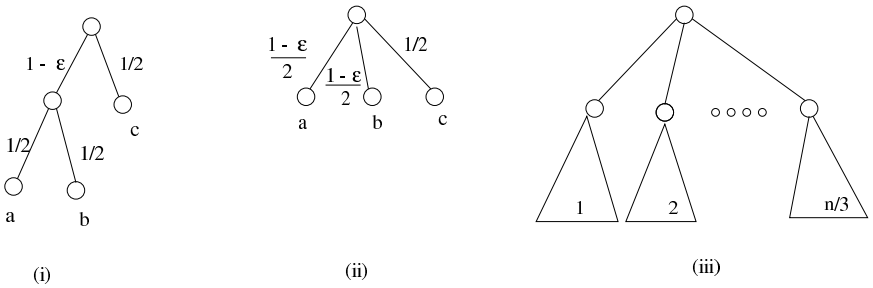

Figure 4. Topologies used in the lower bound proof – in (iii) each of the $n/3$ subtrees are identical to tree (i) or tree (ii)

1 for the trees (i) and (ii) respectively. In order to infer the topology of these trees, an algorithm has to estimate $\alpha(a,b)$ with enough precision to classify it correctly (as either 1 or $1-\epsilon$). This is the same problem as the one in Lemma 1 in Section 5.1. So by Corollary 2, $m = \Omega\left(\frac{1}{\epsilon}\left(\ln\frac{\epsilon}{\delta}\right)\right)$ probe packets are required to solve this problem. If all the edge weights are to be estimated as well, then by Corollary 3, $m = \Omega\left(\frac{1}{\epsilon^2}\left(\ln\frac{\epsilon}{\delta}\right)\right)$ probe packets are required
The same argument can be extended for the more general tree shown in figure 4 (iii). Here, instead of distinguishing between 2 possible trees, an algorithm has to classify correctly each of $\frac{n}{3}$ trees. Thus, there are $\frac{n}{3}$ pairs of nodes (a,b) for which $\alpha(a,b)$ must be determined. Also, each of the $\frac{n}{3}$ trees are disjoint and therefore the problem of classifying each is probabilistically independent of the others.
For the maximum failure probability of the algorithm to be δ, the maximum failure probability for any of these $\frac{n}{3}$ pairs must be at most $3\delta/n$. So by substituting $3\delta/n$ for δ in Lemma 1 and Corollaries 2 and 3, we get the following.

Lemma 4 *The number of samples, m, required by any algorithm to infer the topology of a multicast tree with n leaves with failure probability at most $\delta > 0$ satisfies*

$$m = \Omega\left(\frac{1}{\epsilon}\left(\ln\frac{n}{\delta}\right)\right)$$

Lemma 5 *The number of samples, m, required by any algorithm to infer both the topology and the edge costs of a multicast tree with n leaves with failure probability at most $\delta > 0$ satisfies*

$$m = \Omega\left(\frac{1}{\epsilon^2}\left(\ln\frac{n}{\delta}\right)\right)$$

Next we prove upper bounds on the sample complexity.

5.3 Upper bounds for a single edge

Lemma 6 *The number of probes, n, required to distinguish between two edges with weights 0.5 and $0.5+\epsilon$ with error probability at most $\delta > 0$ satisfies*

$$n = O\left(\frac{1}{\epsilon^2}\left(\ln\frac{1}{\delta}\right)\right)$$

Proof: In order to find an upper bound on the number of probes required to ensure an error probability of at most δ, we can set the upper bound on the probability $\mathrm{Prob}(S/n > (1+\epsilon)/2)$ to be at most δ. Since we know from Equation 8 in Theorem 12 that

$$\begin{aligned}\mathrm{Prob}(S/n > (1+\epsilon)/2) &= \mathrm{Prob}(S > (1+\epsilon)n/2)\\ &\le \frac{C}{\sqrt{n}}\exp(-n\mathbb{I}((1+\epsilon)/2),\end{aligned}$$

where $\mathbb{I}()$ is defined as $\mathbb{I}(x) = x\ln\frac{x}{p} + (1-x)\ln\frac{1-x}{1-p}$ (see Equation 4 in the Appendix), therefore,

$$\frac{C}{\sqrt{n}}\exp\left(-n\mathbb{I}\left(\frac{(1+\epsilon)}{2}\right)\right) \le \delta, \tag{6}$$

where $C = \frac{0.69(1-p)}{x-p}\sqrt{\frac{x}{1-x}} = \frac{0.69}{\epsilon}\sqrt{\frac{1+\epsilon}{1-\epsilon}}$ for $x = \frac{1+\epsilon}{2}$ and $p = \frac{1}{2}$. Squaring both sides of inequality 6, we have $\frac{C^2}{n}\exp\left(-2n\mathbb{I}\left(\frac{(1+\epsilon)}{2}\right)\right) \le \delta^2$ which can be rewritten as $2n\mathbb{I}\left(\frac{(1+\epsilon)}{2}\right)\exp\left(2n\mathbb{I}\left(\frac{(1+\epsilon)}{2}\right)\right) \ge 2\mathbb{I}\left(\frac{(1+\epsilon)}{2}\right)\frac{C^2}{\delta^2}$. Using Lemma 15, it follows that

$$2n\mathbb{I}\left(\frac{(1+\epsilon)}{2}\right) \le \ln\left(\frac{2A}{\ln A}\right)$$

where $A = 2\mathbb{I}\left(\frac{(1+\epsilon)}{2}\right)\frac{C^2}{\delta^2}$. So,

$$\begin{aligned}n &\le \frac{\ln\left(\frac{2A}{\ln A}\right)}{2\mathbb{I}\left(\frac{(1+\epsilon)}{2}\right)}\\ &\le \frac{\ln\left(\frac{2A}{\ln A}\right)}{2\epsilon^2(1-\epsilon/2)} \text{ by Lemma 17}\\ &\le \frac{\ln 2A}{\epsilon^2(2-\epsilon)}\\ &\le \frac{\ln 2A}{\epsilon^2}\\ &= \frac{\ln 4\mathbb{I}\left(\frac{(1+\epsilon)}{2}\right)\frac{C^2}{\delta^2}}{4\epsilon^2}\\ &\le \frac{1}{4\epsilon^2}\left(\ln\frac{4\epsilon^2C^2}{\delta^2}\right) \text{ by Lemma 17}\\ &= O\left(\frac{1}{\epsilon^2}\left(\ln\frac{1}{\delta}\right)\right)\end{aligned}$$

The last statement follows from the fact that $(\epsilon C)^2 = (0.69)^2 \frac{1+\epsilon}{1-\epsilon} = O(1)$. ■

5.4 Upper bounds for algorithm TD

The upper bounds presented below assume that the fraction of probe packets reaching any receiver is bounded below by a constant. Our bounds are functions of the inverse of these fractions, but the asymptotic notation hides these constants. We could parameterize this dependence, but chose not to, since any network where the number of packets reaching the leaves is very low is not very useful in practice.

Observation 1 *Any tree with n leaves has at most $2n-2$ edges.*

Theorem 7 *Using $T = \Theta\left(\frac{1}{\epsilon^2}\left(\ln \frac{n}{\delta}\right)\right)$ probe packets, each of the $p(i,j)$ values are computed to within an error of $\frac{\epsilon}{2}$ with probability $1 - \frac{\delta}{2n}$.*

Proof: We wish to estimate the logical multicast tree with probability $1-\delta$, i.e. the algorithm may fail to estimate the correct tree with probability at most δ. So it suffices to compute the weight of every edge within an error of $\frac{\epsilon}{2}$ with probability at least $1-\frac{\delta}{2n}$. The probability of algorithm TD failing to compute the $p(i,j)$ value within an error of $\frac{\epsilon}{2}$ for any edge is at most $\frac{\delta}{2n}$, and so the probability of failing on any of the $2n-2$ edges is no more than δ. So, substituting $\frac{\epsilon}{2}$ for ϵ and $\frac{\delta}{2n}$ for δ in Lemma 6, we have the result. ■

5.5 Running time of TD (centralized)

The sample complexity of algorithm TD is $T = \frac{2}{\epsilon^2} \ln \frac{n}{\delta}$.
Since, there are $3n^2$ parameters (viz. $\alpha(i,j), \beta(i,j), \gamma(i,j)$) that the algorithm initially computes from the T samples, the time complexity is $O(n^2 T) = O(\frac{n^2}{\epsilon^2} \ln \frac{n}{\delta})$.
The computation of the $\alpha(i,j), \beta(i,j), \gamma(i,j)$ parameters from the samples can be expressed as a matrix multiplication. This leads to the following observation.

Theorem 8 *Using fast matrix multiplication algorithms one can reduce the time taken to compute the α parameters to $O(n^{1.376}T)$.*

6 Discussion

In this paper, we presented a top-down algorithm for loss-based multicast tree construction. We described a distributed version of the algorithm that can scale to very large networks. We also analyzed the algorithms and proved tight bounds on its sample complexity. Our bounds describe how the sample complexity scales with different parameters, including the fact that the number of probe packets required to estimate the topology is proportional to $1/\epsilon$. Another interesting aspect of our analysis is that we show that the number of probes required to estimate the weights of the multicast tree edges is proportional to $1/\epsilon^2$. These results confirm the intuition that higher numbers of probes are needed for reliable tree estimation in low-error networks, and suggest that this technique may require unrealistically large numbers of probe packets in networks with very low congestion.

Acknowledgments: We thank Micah Adler for his valuable suggestions.

References

[1] The network simulator: ns-2. Available at `http://www.isi.edu/nsnam/ns/`.

[2] W. Bryc. Elementary large deviations with applications. Lecture Notes, University of Cincinnati, available online at `http://math.uc.edu/~brycw/classes/576/`.

[3] R. Caceres, N. Duffield, J. Horowitz, F. L. Presti, and D. Towsley. Loss-based inference of multicast network topology. In *IEEE CDC*, 1999.

[4] R. Castro, M. J. Coates, and R. Nowak. Likelihood-based hierarchical clustering. *IEEE Transactions on Signal Processing*, 52:2308–2321, August 2004.

[5] H. Chernoff. A measure of asymptotic efficiency for test of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–507.

[6] M. Coates, A. Hero, R. Nowak, and B. Yu. Internet tomography. *IEEE Signal Processing Magazine*, May 2002.

[7] M. J. Coates and R. Nowak. Network delay distribution inference from end-to-end unicast measurement. In *Proceedings of IEEE International Acoustic, Speech, and Signal Processing*, 2001.

[8] A. Dembo and O. Zeitouni. *Large Deviations Techniques and Applications*. Springer Verlag, 2 edition, 1998.

[9] N. Duffield, J. Horowitz, D. Towsley, W. Wei, and T. Friedman. Multicast-based loss inference with missing data. *IEEE Journal of Selected Areas of Communications*, 20(4):700–713, May 2002.

[10] S. Floyd, V. Jacobson, S. C. Liu, McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997.

[11] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic press, New York, 1990.

[12] S. Goldman. *Computational Learning Theory*. CRC Press, 1999.

[13] I. Kouvelas, V. Hardman, and J. Crowcroft. Network adaptive continuous-media applications through self organised transcoding. In *Proceedings of the Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 1998.

[14] J. C. Lin and S. Paul. RMTP:a reliable multicast transport protocol. In *Proceedings of IEEE INFOCOM*, pages 1414–1424, March 1996.

[15] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[16] M. Rabbat, R. Nowak, and M. Coates. Multiple source, multiple destination network tomography. In *Proceedings of IEEE INFOCOM*, 2004.

[17] S. Ratnasamy and S. McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *Proceedings of IEEE INFOCOM*, March 1999.

[18] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement. *IEEE/ACM Transactions on Networking (TON)*, 10(3):381–395, 2002.

[19] A. Weiss. An introduction to large deviations for communication networks. *IEEE Journal on Selected Areas in Communications*, 13(6):938–952, August 1995.

A Bounds for Binomial probabilities

In this section, we obtain fairly tight tail bounds for random variables that have the Binomial distribution. While these results are elementary and have been known before, the only place we found them was in some online course notes [2]. We have followed most of their notation and have corrected some minor errors.

Let $X = Bin(n,p)$ be a random variable with the Binomial distribution. That is

$$\Pr(X=k) = \binom{n}{k} p^k (i-p)^{n-k}.$$

Now we derive tight founds for the tail of X. The bounds are in terms of a rate function $\mathbb{I}(x)$ which was defined as follows.

$$\mathbb{I}(x) = x \ln \frac{x}{p} + (1-x) \ln \frac{1-x}{1-p} \tag{7}$$

We begin by stating some simple properties of factorials and binomial coefficients.

Lemma 9

$$\left(\frac{2e}{3}\right)^{1.5} \left(\frac{n}{e}\right)^n \leq \frac{n!}{\sqrt{n}} \leq e \left(\frac{n}{e}\right)^n$$

or equivalently,

$$2.4 n^{n+0.5} e^{-n} \leq n! \leq 2.8 n^{n+0.5} e^{-n}$$

Lemma 10

$$\begin{aligned} & \frac{0.3}{\sqrt{n}} \frac{1}{\sqrt{(1-\frac{k}{n})\frac{k}{n}}} \left[\left(\frac{p}{k/n}\right)^{k/n} \left(\frac{(1-p)}{1-k/n}\right)^{1-k/n} \right]^n \\ \leq \; & \Pr(X=k) \leq \\ & \frac{0.49}{\sqrt{n}} \frac{1}{\sqrt{(1-\frac{k}{n})\frac{k}{n}}} \left[\left(\frac{p}{k/n}\right)^{k/n} \left(\frac{(1-p)}{1-k/n}\right)^{1-k/n} \right]^n \end{aligned}$$

Proof: The second inequality can be proved as follows.

$$\begin{aligned} & \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k} \\ \leq \; & \frac{2.8 n^{n+0.5} e^{-n}}{2.4^2 k^{k+0.5} e^{-k} (n-k)^{n-k+0.5} e^{-n+k}} p^k (1-p)^{n-k} \\ = \; & \frac{2.8 n^{n+0.5}}{2.4^2 k^{k+0.5} (n-k)^{n-k+0.5}} p^k (1-p)^{n-k} \\ = \; & \frac{2.8}{2.4^2} \left(\frac{n}{(n-k)k}\right)^{0.5} \left(\frac{np}{k}\right)^k \left(\frac{n(1-p)}{n-k}\right)^{n-k} \\ = \; & \frac{2.8}{2.4^2 \sqrt{n}} \left(\frac{1}{(1-\frac{k}{n})\frac{k}{n}}\right)^{0.5} \left(\frac{p}{k/n}\right)^k \left(\frac{(1-p)}{1-k/n}\right)^{n-k} \\ = \; & \frac{2.8}{2.4^2 \sqrt{n}} \frac{1}{\sqrt{(1-\frac{k}{n})\frac{k}{n}}} \left[\left(\frac{p}{k/n}\right)^{\frac{k}{n}} \left(\frac{(1-p)}{1-\frac{k}{n}}\right)^{1-\frac{k}{n}} \right]^n \\ \leq \; & \frac{0.49}{\sqrt{n}} \frac{1}{\sqrt{(1-\frac{k}{n})\frac{k}{n}}} \left[\left(\frac{p}{k/n}\right)^{k/n} \left(\frac{(1-p)}{1-\frac{k}{n}}\right)^{1-\frac{k}{n}} \right]^n \end{aligned}$$

The first inequality can be proved as follows.

$$\begin{aligned} & \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k} \\ \geq \; & \frac{2.4 n^{n+0.5} e^{-n}}{2.8^2 k^{k+0.5} e^{-k} (n-k)^{n-k+0.5} e^{-n+k}} p^k (1-p)^{n-k} \\ = \; & \frac{2.4 n^{n+0.5}}{2.8^2 k^{k+0.5} (n-k)^{n-k+0.5}} p^k (1-p)^{n-k} \\ = \; & \frac{2.4}{2.8^2} \left(\frac{n}{(n-k)k}\right)^{0.5} \left(\frac{np}{k}\right)^k \left(\frac{n(1-p)}{n-k}\right)^{n-k} \\ = \; & \frac{2.4}{2.8^2 \sqrt{n}} \left(\frac{1}{(1-\frac{k}{n})\frac{k}{n}}\right)^{0.5} \left(\frac{p}{k/n}\right)^k \left(\frac{(1-p)}{1-k/n}\right)^{n-k} \\ = \; & \frac{2.4}{2.8^2 \sqrt{n}} \frac{1}{\sqrt{(1-\frac{k}{n})\frac{k}{n}}} \left[\left(\frac{p}{k/n}\right)^{k/n} \left(\frac{(1-p)}{1-\frac{k}{n}}\right)^{1-\frac{k}{n}} \right]^n \\ \geq \; & \frac{0.3}{\sqrt{n}} \frac{1}{\sqrt{(1-\frac{k}{n})\frac{k}{n}}} \left[\left(\frac{p}{k/n}\right)^{k/n} \left(\frac{(1-p)}{1-\frac{k}{n}}\right)^{1-\frac{k}{n}} \right]^n \end{aligned}$$

■

Lemma 11

$$\exp(-\mathbb{I}(x)) = \left(\frac{p}{x}\right)^x \left(\frac{1-p}{1-x}\right)^{1-x}$$

where $\mathbb{I}(x)$ is defined in Equation 4.

Proof: $\mathbb{I}(x) = x \ln \frac{x}{p} + (1-x) \ln \frac{1-x}{1-p}$. Therefore, $\exp(-\mathbb{I}(x)) = \left(\frac{p}{x}\right)^x \left(\frac{1-p}{1-x}\right)^{1-x}$ ■

Theorem 12 *If X is Binomial Bin(n,p) then for all $x > p$ and $n > 2/(1-x)$*

$$\Pr(X > nx) \leq \frac{C}{\sqrt{n}} \exp(-n\mathbb{I}(x)) \tag{8}$$

where $C = C(x,p) = \frac{0.69(1-p)}{x-p} \sqrt{\frac{x}{1-x}}$

Moreover, for all $x > p$ and $n > \frac{1}{x-p}$ we have

$$\Pr(X > nx) \geq \frac{c}{\sqrt{n}} \exp(-n\mathbb{I}(x)) \tag{9}$$

where $c = c(x,p) = \frac{0.15p(1-x)}{\sqrt{x}(1-p)^{3/2}}$.
Thus, for all $x < p$ and n large enough we have

$$\frac{c}{\sqrt{n}} \exp(-n\mathbb{I}(x)) \leq \Pr(X < nx) \leq \frac{C}{\sqrt{n}} \exp(-n\mathbb{I}(x))$$

We note that $\mathbb{I}(x)$ is a convex function with the unique minimum at $x = p$. Indeed, the derivatives are $\mathbb{I}'(x) = \ln\frac{x}{1-x} - \ln\frac{p}{1-p}$, $\mathbb{I}''(x) = \frac{1}{x(1-x)} > 0$. This implies that $\mathbb{I}(x)$ is an increasing function for $x > p$.

Proof: To prove the lower bound 9 let $k = \lfloor xn \rfloor$. Since $n > \frac{1}{x-p}$ we have $p < x - \frac{1}{n} < \frac{k}{n} \leq x$. In particular, since $\mathbb{I}(x)$ is increasing for $x > p$, we have

$$\mathbb{I}\left(\frac{k}{n}\right) \leq \mathbb{I}(x) \tag{10}$$

Notice that

$$\begin{aligned}
\frac{\Pr(X=k+1)}{\Pr(X=k)} &= \frac{C(n,k+1)p^{k+1}(1-p)^{n-k-1}}{C(n,k)p^k(1-p)^{n-k}} \\
&= \frac{(n-k)p}{(k+1)(1-p)} \\
&\geq \frac{p}{1-p}\frac{1-\frac{k}{n}}{1+\frac{k}{n}} \\
&\geq \frac{p}{1-p}\frac{1-x}{1+x} \\
&\geq \frac{p}{2}\frac{1-x}{1-p} \qquad (11)
\end{aligned}$$

The lower bound can be derived from 10, 11, the fact that $\Pr(X > nx) \geq \Pr(X = k+1)$ and inequalities in Lemma 10. Namely,

$$\begin{aligned}
\Pr(X=k) &\geq \frac{0.3}{\sqrt{n}} \frac{1}{\sqrt{\frac{k}{n}(1-\frac{k}{n})}} \left[\left(\frac{p}{\frac{k}{n}}\right)^{\frac{k}{n}} \left(\frac{1-p}{1-\frac{k}{n}}\right)^{1-\frac{k}{n}}\right]^n \\
&= \frac{0.3}{\sqrt{n}} \frac{1}{\sqrt{\frac{k}{n}(1-\frac{k}{n})}} \exp\left(-n\mathbb{I}\left(\frac{k}{n}\right)\right) \\
&\geq \frac{0.3}{\sqrt{n}} \frac{1}{\sqrt{x(1-p)}} \exp(-n\mathbb{I}(x)) \\
\Pr(X > nx) &\geq \Pr(X = k+1) \\
&\geq \frac{p}{2}\frac{1-x}{1-p} \Pr(X=k) \\
&\geq \frac{p}{2}\frac{1-x}{1-p}\frac{0.3}{\sqrt{n}} \frac{1}{\sqrt{x(1-p)}} \exp(-n\mathbb{I}(x)) \\
&= \frac{0.15p(1-x)}{\sqrt{x}(1-p)^{1.5}} \frac{1}{\sqrt{n}} \exp(-n\mathbb{I}(x))
\end{aligned}$$

To prove the upper bound, we will show that $\Pr(X > nx)$is within a multiplicative factor of $\Pr(X = k+1)$ for $k = \lfloor nx \rfloor$. To see this notice that (as in 11) we have

$$\frac{\Pr(X=j+1)}{\Pr(X=j)} = \frac{p}{1-p}\frac{n-j}{j+1} \leq \frac{p}{1-p}\frac{n-k}{k+1}$$

for all $j \geq k$.
Therefore for $j \geq 0$

$$\begin{aligned}
&\frac{\Pr(X=k+j)}{\Pr(X=k)} \\
&= \frac{\Pr(X=k+1)}{\Pr(X=k)}\frac{\Pr(X=k+2)}{\Pr(X=k+1)} \cdots \frac{\Pr(X=j)}{\Pr(X=j-1)} \\
&\leq \left(\frac{p}{1-p}\frac{n-k}{k+1}\right)^j
\end{aligned}$$

Let $r = \frac{p}{1-p}\frac{n-k-1}{k+2}$ and put $k = \lfloor nx \rfloor$. Since $\frac{k+2}{n+2} > \frac{k}{n} > p$ we have $r < 1$. Therefore

$$\begin{aligned}
\Pr(X > nx) &= \sum_{j=k+1}^{n} \Pr(X=j) \\
&\leq \Pr(X=k+1)\sum_{j=0}^{\infty} r^j \\
&= \Pr(X=k+1)\frac{(1-p)(k+2)}{k+2-(n+1)p} \\
&= \Pr(X=k+1)(1-p)\frac{x'}{x'-p}
\end{aligned}$$

where $x' = \frac{k+2}{n+1} \geq \frac{nx+1}{n+1} \geq x > p$. In particular, since $x \mapsto \frac{x}{x-p} = 1 + \frac{p}{x-p}$ is a decreasing function of x, it follows that

$$\Pr(X > nx) \leq \Pr(X=k+1)(1-p)\frac{x}{x-p}$$

Now we proceed similarly as in the proof of the lower bound.

$$\begin{aligned}
&\Pr(X=k+1) \\
&\leq \frac{0.49}{\sqrt{n}} \frac{1}{\sqrt{\frac{k+1}{n}(1-\frac{k+1}{n})}} \left[\left[\frac{p}{\frac{k+1}{n}}\right]^{\frac{k+1}{n}} \left[\frac{1-p}{1-\frac{k+1}{n}}\right]^{1-\frac{k+1}{n}}\right]^n \\
&= \frac{0.49}{\sqrt{n}} \frac{1}{\sqrt{\frac{k+1}{n}(1-\frac{k+1}{n})}} \exp\left(-n\mathbb{I}\left(\frac{k+1}{n}\right)\right) \\
&\leq \frac{0.49}{\sqrt{n}} \frac{1}{\sqrt{x}\sqrt{1-\frac{1}{n}-x}} \exp(-n\mathbb{I}(x))
\end{aligned}$$

where in the last bound we used the fact that $\mathbb{I}(\cdot)$ in increasing on $(p,1)$, and $\frac{k+1}{n} > x$. Therefore,

$$\begin{aligned}
&\Pr(X > nx) \\
&\leq (1-p)\frac{x}{x-p} \Pr(X=k+1) \\
&\leq (1-p)\frac{x}{x-p}\frac{0.49}{\sqrt{n}} \frac{1}{\sqrt{x}\sqrt{1-\frac{1}{n}-x}} \exp(-n\mathbb{I}(x))
\end{aligned}$$

$$
\begin{aligned}
&= (1-p)\frac{0.49x}{x-p}\sqrt{\frac{1-x}{1-\frac{1}{n}-x}}\frac{1}{\sqrt{x(1-x)}}\frac{\exp(-n\mathbb{I}(x))}{\sqrt{n}} \\
&\leq (1-p)\frac{0.49x}{x-p}\sqrt{2}\frac{1}{\sqrt{x(1-x)}}\frac{\exp(-n\mathbb{I}(x))}{\sqrt{n}} \\
&\quad \text{since } 1-x-\frac{1}{n} < \frac{1-x}{2} \text{ for } n > \frac{2}{1-x} \\
&\leq \frac{0.693(1-p)}{x-p}\sqrt{\frac{x}{1-x}}\frac{1}{\sqrt{n}}\exp(-n\mathbb{I}(x))
\end{aligned}
$$

■

It is worth noting that the upper bound proved in the previous theorem is stronger than the bound in the well-known Chernoff Bound [5, 15].

B Some useful Lemmas

We present some lemmas used in our proofs.

Lemma 13 *Suppose $A > 0$. Then*

$$\ln\left(\frac{A}{\ln A}\right) > \frac{1}{2}\ln A.$$

Lemma 14 *Suppose $\delta < 0.07\epsilon$, and $A = 2\mathbb{I}\left(p+\frac{\epsilon}{2}\right)\frac{\epsilon^2}{\delta^2}$. Then $A > e$.*

Proof: From Lemma 17, we have $\mathbb{I}\left(p+\frac{\epsilon}{2}\right) > \frac{\epsilon^2}{16p(1-p)}$. Therefore, $A > \frac{\epsilon^2 c^2}{8p(1-p)\delta^2} = \frac{\epsilon^2}{8p(1-p)\delta^2}\frac{0.0225p^2(1-p-\epsilon/2)^2}{(p+\epsilon/2)(1-p)^3}$ $= \frac{0.0225\epsilon^2 p(1-p-\epsilon/2)^2}{(p+\epsilon/2)(1-p)^4\delta^2} > \frac{0.0225\epsilon^2\epsilon(1-1.5\epsilon)^2}{1.5\epsilon(1-\epsilon)^4\delta^2}$. So $A > e$, if $\delta < \sqrt{\frac{0.00552\epsilon^2(1-1.5\epsilon)^2}{(1-\epsilon)^4}} = \frac{0.0743\epsilon(1-1.5\epsilon)}{(1-\epsilon)^2} = 0.07\epsilon\left(\frac{(1-1.5\epsilon)}{(1-\epsilon)^2}\right)$. ■

Lemma 15 *Suppose $te^t = A > e$. Then*

$$\ln\left(\frac{A}{\ln A}\right) \leq t \leq \ln\left(\frac{2A}{\ln A}\right).$$

Proof: First, note that te^t is a monotonically increasing function. If $t = \ln\left(\frac{A}{\ln A}\right)$, then
$te^t = \left(\frac{A}{\ln A}\right)\ln\left(\frac{A}{\ln A}\right) = A\left(\frac{\ln A - \ln\ln A}{\ln A}\right) \leq A$.
If $t = \ln\left(\frac{2A}{\ln A}\right)$, then $te^t = \left(\frac{2A}{\ln A}\right)\ln\left(\frac{2A}{\ln A}\right) = A\left(\frac{2(\ln 2A - \ln\ln A)}{\ln A}\right) \geq A$, because
$2(\ln 2A - \ln\ln A) \geq \ln A$ when $\ln 2\sqrt{A} \geq \ln\ln A$ or $2\sqrt{A} \geq \ln A$ or $A \geq \left(\frac{\ln A}{2}\right)^2$,
which is true for all $A \geq 1$. ■

Lemma 16 *If $0 < y < 1$, then*

$$\ln(1-y/2) \geq -y(1+y)/2$$

Proof: It suffices to prove that $e^{-y(1+y)/2} \leq 1 - y/2$. However, since $e^{-x} \leq 1 - x + x^2/2$,

$$
\begin{aligned}
e^{-y(1+y)/2} &\leq 1 - y(1+y)/2 + y^2(1+y)^2/8 \\
&\leq 1 - y/2, \text{ if} \\
y^2/2 &\geq y^2(1+y)^2/8 \\
4 &\geq (1+y)^2 \text{ which is true, since } y \leq 1
\end{aligned}
$$

■

Lemma 17

$$\frac{\epsilon^2}{16p(1-p)} \leq \mathbb{I}(p+\epsilon/2) \leq \frac{\epsilon^2}{4p(1-p)}$$

Proof:

$$
\begin{aligned}
&\mathbb{I}\left(p+\frac{\epsilon}{2}\right) \\
&= \left(p+\frac{\epsilon}{2}\right)\ln\left(\frac{p+\epsilon/2}{p}\right) \\
&\quad + \left(1-p-\frac{\epsilon}{2}\right)\ln\left(\frac{1-p-\epsilon/2}{1-p}\right) \\
&= \left(p+\frac{\epsilon}{2}\right)\ln\left(1+\frac{\epsilon}{2p}\right) \\
&\quad + \left(1-p-\frac{\epsilon}{2}\right)\ln\left(1-\frac{\epsilon}{2(1-p)}\right)
\end{aligned}
$$

Since $\ln(1+x) < x$ and $\ln(1-x) < -x$, therefore

$$
\begin{aligned}
&\mathbb{I}\left(p+\frac{\epsilon}{2}\right) \\
&\leq \left(p+\frac{\epsilon}{2}\right)\frac{\epsilon}{2p} - \left(1-p-\frac{\epsilon}{2}\right)\frac{\epsilon}{2(1-p)} \\
&= \frac{\epsilon^2}{4p} + \frac{\epsilon^2}{4(1-p)} \\
&= \frac{\epsilon^2}{4p(1-p)}
\end{aligned}
$$

and since $\ln(1+x) > x(1-x/2)$ and $\ln(1-y/2) > -y(1+y)/2$, therefore

$$
\begin{aligned}
&\mathbb{I}\left(p+\frac{\epsilon}{2}\right) \\
&\geq \left(p+\frac{\epsilon}{2}\right)\frac{\epsilon}{2p}\left(1-\frac{\epsilon}{4p}\right) - \\
&\quad \left(1-p-\frac{\epsilon}{2}\right)\frac{\epsilon}{2(1-p)}\left(1+\frac{\epsilon}{(1-p)}\right) \\
&= \frac{\epsilon}{2}\left[\left(1+\frac{\epsilon}{2p}\right)\left(1-\frac{\epsilon}{4p}\right) - \right. \\
&\quad \left.\left(1-\frac{\epsilon}{2(1-p)}\right)\left(1+\frac{\epsilon}{(1-p)}\right)\right] \\
&= \frac{\epsilon}{2}\left[\left(1+\frac{\epsilon}{4p}-\frac{\epsilon^2}{8p^2}\right) - \right. \\
&\quad \left.\left(1+\frac{\epsilon}{2(1-p)}-\frac{\epsilon^2}{2(1-p)^2}\right)\right]
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{\epsilon}{2}\left[\frac{\epsilon}{4p} - \frac{\epsilon^2}{8p^2} - \frac{\epsilon}{2(1-p)} + \frac{\epsilon^2}{2(1-p)^2}\right] \\
&\geq \frac{\epsilon^2}{4p(1-p)}\frac{1+p}{2} - \frac{\epsilon^3}{16p^2} \\
&\geq \frac{\epsilon^2}{4p(1-p)}\left[\frac{1+p}{2} - \frac{\epsilon(1-p)}{4p}\right] \\
&\geq \frac{\epsilon^2}{16p(1-p)} \text{ since } p > \epsilon
\end{aligned}
$$

■