Network monitoring using Tomography

[1]Devashish Patel [2]Rohan Rambhia [3]Amit Rander and [4]Kailas Devadkar

***Abstract*-- Today's Internet is a massive, distributed network which continues to explode in size as e commerce and related activities grow. The heterogeneous and largely unregulated structure of the Internet renders tasks like dynamic routing, optimized service provision, service level verification and detection of anomalous/malicious behavior extremely challenging. The problem is compounded by the fact that one cannot rely on the cooperation of individual servers and routers to aid in the collection of network traffic measurements vital for these tasks. This paper introduces network tomography, a new field which we believe will benefit greatly from the wealth of statistical theory and algorithms. It focuses on the application of pseudo-likelihood methods and tree estimation formulations for network monitoring.**

***Index Terms*-- Network tomography, pseudo-likelihood, topology identification, tree estimation.**

I. Introduction

LARGE-SCALE network inference problems can be classified according to the type of data acquisition and the performance parameters of interest. To discuss these distinctions, we require some basic definitions. Consider the network depicted in Fig. 1. Each node represents a computer terminal, router or sub-network (consisting of multiple computers/routers). A connection between two nodes is called a path. Each path consists of one or more links—direct connections with no intermediate nodes. The links may be unidirectional or bidirectional, depending on the level of abstraction and the problem context. Each link can represent a chain of physical links connected by intermediate routers. Messages are transmitted by sending *packets* of bits from a *source* node to a *destination* node along a path which generally passes through several other nodes. Broadly speaking, large scale network inference involves estimating network performance parameters based on traffic measurements at a limited subset of the nodes. Vardi (1996) was one of the first researchers to rigorously

study this sort of problem and he coined the term *network tomography* due to the similarity between network inference and medical tomography. Two forms of network tomography have been addressed in the recent literature: (1) link-level parameter estimation based on end-to-end, path-level traffic measurements [1] and (2) sender–receiver path-level traffic intensity estimation based on link-level traffic measurements [2]. In link-level parameter estimation, the traffic measurements typically consist of counts of packets transmitted and/or received between source and destination nodes or time delays between packet transmissions and receptions. The goal is to estimate the loss rate or the queuing delay on each link. The measured time delays are due to both propagation delays and router processing delays along the path. The path delay is the sum of the delays on the links that comprise the path; the link delay comprises both the propagation delay on that link and the queuing delay at the routers that lie along that link. A packet is dropped if it does not successfully reach the input buffer of the destination node. Link delays and occurrences of dropped packets are inherently random. Random link delays can be caused by router output buffer delays, router packet servicing delays and propagation delay variability.

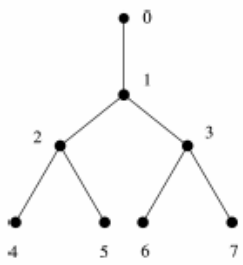

Fig. 1. An arbitrary virtual multicast tree with four receivers

Dropped packets on a link are usually due to overload of the finite output buffer of one of the routers encountered when traversing the link, but may also be caused by equipment downtime due to maintenance or power failures. Random link delays and packet losses become particularly substantial when there is a large amount of cross traffic competing for service by routers along a path. In path-level traffic intensity estimation, the measurements consist of counts of packets that pass through nodes in the network. In privately owned networks, the collection of such measurements is relatively straightforward. Based on these measurements, the goal is to estimate how much traffic originated from a specified node and was destined for a specified receiver. The combination of the traffic intensities of all these origin– destination pairs forms the origin–destination traffic matrix. In this problem not only are the node-level measurements inherently random, but the parameter to be estimated (the origin–destination traffic

[1]Devashish Patel is a student of Sardar Patel College of Engineering (IT Department) , Mumbai, India(email : devshish_p@yahoo.com).

[2]Rohan Rambhia is a student of Sardar Patel College of Engineering (IT Department) , Mumbai, India(email rohan_rambhia@yahoo.com).

[3]Amit Rander is a student of Sardar Patel College of Engineering (IT Department) , Mumbai, India(email amitrander@yahoo.com).

[4]Kailas Devadkar is a lecturer of Sardar Patel Institute of Technology (IT Department) , Mumbai, India(email : kailaskd@rediffmail.com)

matrix) must itself be treated not as a fixed parameter, but as a random vector. Randomness arises from the traffic generation itself, rather than perturbations or measurement noise. The inherent randomness in both link-level and path level measurements motivates the adoption of statistical methodologies for large-scale network inference and tomography. Many network tomography problems can be roughly approximated by the (not necessarily Gaussian) linear model

$$Y_t = AX_t + \varepsilon \tag{1}$$

where Y_t is a vector of measurements (e.g., packet counts or end-to-end delays) recorded at a given time t at a number of different measurement sites, A is a *routing matrix*, ε is a noise vector and X_t is a vector of time-dependent packet parameters (e.g., mean delays, logarithms of packet transmission probabilities over a link or the random origin–destination traffic vector). In some cases the vector X_t is a random vector with an underlying parameterized distribution $f(X_t \mid \theta_t)$, and it is the parameters θ_t that interest us. Typically, but not always, A is a binary matrix (the *i, j* th element is equal to 1 or 0) that captures the topology of the network. In this paper, we consider the problems of using the observations Yt to estimate θ_t, X_t or A (see Section 4). What sets the large-scale network inference problem (1) apart from other network inference problems is the potentially very large dimension of A which can range from a half a dozen rows and columns for a few packet parameters and a few measurement sites in a small local area network, to thousands or tens of thousands of rows and columns for a moderate number of parameters and measurements sites in the Internet. The associated high-dimensional problems of estimating X_t are specific examples of *inverse problems*. Inverse problems have a very extensive literature [14]. Solution methods for such inverse problems depend on the nature of the noise ε and the A matrix, and typically require iterative algorithms since they cannot be solved directly. In general, A is not full rank, so that identify-ability concerns arise. Either one must be content to resolve only linear combinations of the parameters or one must employ statistical means to introduce regularization and induce identify-ability. In most of the large-scale Internet inference and tomography problems studied to date, the components of the noise vector ε are assumed to be approximately independent Gaussian, Poisson, binomial or multinomial distributed. When the noise is Gaussian distributed with covariance independent of AX_t, methods such as recursive linear least squares can be implemented using conjugate gradient, Gauss–Seidel and other iterative equation solvers. When the noise is modeled as Poisson, binomial or multinomial distributed, more sophisticated statistical methods, such as reweighed nonlinear least squares, maximum likelihood via expectation–maximization (EM) and maximum a posteriori via Markov chain Monte Carlo (MCMC) algorithms, become necessary.

II. Technical Work Preparation

In developing methods to perform network tomography, there is a trade-off between statistical efficiency (accuracy) and computational overhead. In the past, researchers have addressed the extreme computational burden posed by some of the tomographic problems, developing suboptimal but lightweight algorithms, including a fast recursive algorithm for link delay distribution inference in a multicast framework [3] and a method-of-moments approach for origin–destination matrix inference [4]. More accurate but computationally burdensome approaches have also been explored, including maximum-likelihood methods [5], but in general they are too intensive computationally for any network of reasonable scale.

A. Pseudo-Likelihood Approach

A unified pseudo-likelihood approach [6] that eases the computational burden but maintains good statistical efficiency. The idea of modifying likelihood is not new, and many modified likelihood models have been proposed, for example, pseudo-likelihood for Markov random fields, partial likelihood for hazards regression and quasi-maximum likelihood for finance models. In this section, we describe the pseudo-likelihood approach. The network tomography model we consider in this section is a special case of (1), in which the error term Є is omitted for further simplification. Hence the model can be rewritten as

$$Y = AX \tag{2}$$

where $X = (X_1, \ldots, X_J)'$ is a J -dimensional vector of network dynamic parameters (e.g., link delay, traffic flow counts at a particular time interval), $Y = (Y_1, \ldots, Y_I)$ is an I -dimensional vector of measurements and A is an $I \times J$ routing matrix. As mentioned before due to the temporal and spatial correlations between network traffic, but it is a good first-step approximation. Furthermore, we assume that

$$X_j \approx f_j(\theta_j) \tag{3}$$

for j=1…J and where f_j is a density function and θ_j is its parameter. Then the parameter of the whole model is $\theta = (\theta_1, \ldots \theta_J)$. The main idea of the pseudo-likelihood approach is to decompose the original model into a series of simpler sub problems by selecting pairs of rows from the routing matrix A and to form the pseudo-likelihood function by multiplying the marginal likelihoods of such sub problems. Let S denote the set of sub problems by selecting all possible pairs of rows from the routing matrix A: $S = \{s = (i_1, i_2) : 1 \le i_1 < i_2 \le I\}$. Then for each sub problem $s \in S$, we have

$$Y^s = A^s X^s \tag{4}$$

where X^s is the vector of network dynamic components involved in the given sub problem s, A^s is the corresponding sub-routing matrix and $Y^s = (Yi_1, Yi_2)'$ is the observed

measurement vector of s. Let θ^s be the parameter of s and let $p^s(Y^s;\theta^s)$ be its marginal likelihood function. Usually sub problems are dependent, but ignoring such dependencies, the pseudo-likelihood function can be written as the product of marginal likelihood functions of all sub problems, that is, given observation $y_1 \ldots y_T$, the pseudo-log-likelihood function is defined as

$$L^p(y_1 \ldots y_T;\theta) = \sum_{t=1}^{T} \sum_{s \in S} l^s\left(y_s^T;\theta_s\right) \quad (5)$$

where $l^s(Y^s;\theta^s) = \log p^s(Y^s;\theta^s)$ is the log-likelihood function of sub problem$s$. Maximizing the pseudo-log-likelihood function L^p gives the maximum-pseudo likelihood estimate (MPLE) of parameter θ. Maximizing the pseudo-likelihood is not an easy task because $L^p(y_1, \ldots, y_T;\theta)$ is a summation of many functions. Since the maximization of the pseudo-likelihood function is a typical missing value problem, a pseudo-EM algorithm (a variant of the EM algorithm) [6], is employed to maximize the function $L^p(y_1, \ldots, y_T;\theta)$. Let $l^s(\mathbf{X}^s;\theta^s)$ be the log-likelihood function of a sub problem s given the complete data $\mathbf{X}^s$ and let $\theta^{(k)}$ be the estimate of θ obtained in the kth step. The objective function $Q(\theta, \theta^{(k)})$ to be maximized in the $(k+1)$st step of the pseudo-EM algorithm is defined as

$$Q\left(\theta,\theta^{(k)}\right) = \sum_{t=1}^{T} \sum_{s \in S} E_{\theta^{s(k)}} l^s\left((x_t^s;\theta^s) \mid y_t^s\right) \quad (6)$$

which is obtained by assuming the independence of sub problems in the expectation step. The starting point of the pseudo-EM algorithm can be arbitrary, but just as in the EM algorithm, care needs to be taken to ensure that the algorithm does not converge to a local maximum. There are several points worth noting in constructing the pseudo-likelihood function:

1 Selecting three or more rows each time may also be reasonable to construct a pseudo-likelihood function, but there is a trade-off between the computational complexity incurred and the estimation efficiency achieved by taking more dependence structures into account. The experience with the two examples we discuss later shows that selecting two rows each time gives satisfactory estimation results while keeping the computational cost within a reasonable range.

2. Currently all possible pairs are selected to construct the pseudo-likelihood function, but a subset can be judicious chosen to reduce the computation. The pseudo-likelihood is obtained by assuming all subproblems to be independent. Although this assumption is frequently violated, we obtain, under mild conditions, the consistency and asymptotic normality of maximum pseudo-likelihood estimates [6].

In summary, the pseudo-likelihood approach keeps a good balance between the computational complexity and the statistical efficiency of the parameter estimation. Even though the basic idea of divide-and-conquer is not new, it is very powerful when combined with pseudo-likelihood for large network problems.

B. Topology Identification

In the previous section it was assumed that the network topology was known; this knowledge is essential for successful application of the techniques described. When the topology is unknown, tools such as trace route can be used in an attempt to identify it. However, these tools rely on close cooperation from the network internal devices and are incapable of detecting certain types of devices. The tools can thus determine the topology only if the network is functioning properly and network elements are prepared to cooperate and reveal themselves. These conditions are often not met and are becoming more uncommon as the Internet grows in size and speed; there is little motivation for extremely high-speed or heavily loaded switches to spend time processing requests that are not central to the process of communication. Also, the fear of malicious attacks (such as denial of service attacks) forces network administrators to block access to some diagnosis tools on routers (such as ping or the ability to respond to ICMP packets), preventing their use for legitimate purposes. It is therefore desirable to develop a method for estimating topology that uses only measurements taken at the network edge, obtained without cooperation from internal devices. We consider a single source that is communicating with multiple receivers (denote the set of receiver nodes by R). The physical network topology can be represented as a directed graph, where each vertex represents a physical device (e.g., a router or a switch) and the edges correspond to the connections between those devices. In our approach we use only end-to-end measurements and do not use any network device information, which forces us to rely solely on traffic and queuing characteristics.

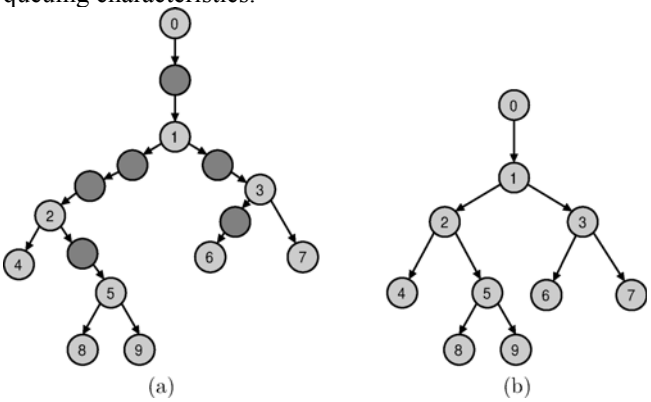

Fig. 2. (a) Physical topology and (b) corresponding logical topology. The darker unnumbered nodes are devices where no branching of traffic occurs and therefore do not appear in the logical topology.

With this limited knowledge, it is only possible to identify the so-called logical topology (see Fig. 2, for an illustration of the distinction between logical and physical topologies). In the logical topology, each vertex represents a physical network device where traffic branching occurs, that is, where two or more source–destination paths diverge. The set of vertices thus corresponds to a subset of the traversed physical devices. An edge is included between two vertices if traffic travels between the corresponding network devices and does not pass through any other devices in the included subset. Each edge corresponds to a connection between two physical devices,

but the connection may include several network devices where no traffic branching occurs. We assume that the routes from the sender to the receivers are fixed during the measurement period, in which case the topology is a tree-structured graph, as in Fig. 2. Every node has at least two children, apart from the root node (which has one) and the leaf nodes (which have none). If all internal nodes have exactly two children, then the tree is called binary. In the following discussion, we focus on the unicast measurement procedure [7] [5] and the hierarchical clustering interpretation of the topology identification problem expounded by [5]. In the topology identification problem the quantity of interest is A, the routing matrix. Note that the entries of this matrix are only 0 or 1. The measurements Y_t are obtained through special measurement techniques described below and the partial ordering of Y_t can be used to determine A. The matrix estimation formulation above is not well suited to the topology identification problem, so we formulate it below as a tree estimation exercise. One can also regard the topology discovery problem as hierarchical clustering. Within such a framework one wants to identify clusters of receivers that share certain properties. In particular, we want to identify the clusters of receiver nodes whose paths from the source node are the same up to a certain point.

Our goal is to identify the logical topology. With each internal node in a tree we associate a metric value γk. We consider only metrics that have a monotonic property: An internal node has a smaller metric value than any of its descendants (e.g., in Fig. 2). Examples of such metrics in networking are the average delay or delay variance experienced by a packet traveling from the source to node *k*. Since we do not know the topology, we cannot estimate the metric values directly, but it is possible to estimate them indirectly. Let a(i, j) denote the nearest common ancestor of a given receiver pair i, j Є R [e.g., a (4, 9) = 2]. Define γij ≡ γa(i,j). The value γij can be regarded as a characterization of the shared portion of the paths from the root to *i* and *j*. The shared path for a pair of nodes *(i, j)* is the path from the root to node *a(i, j)*. In the context of hierarchical clustering, the *γij* can be interpreted as *similarity* values. Note that there is an enforced symmetry in this model: *γij = γji* . Knowledge of the pairwise metric values and the monotonicity property suffices to completely identify the logical topology.

For example, referring to Fig. 2, the metric γ_{i7} is greater than γ_{i7} for all $i \in R \setminus \{6, 7\}$, revealing that nodes 6 and 7 have a common parent in the logical tree. This property can be exploited recursively to devise a simple and effective bottom-up merging algorithm that identifies the complete, logical topology. These same techniques are used in agglomerative hierarchical clustering methods [8] [9] [10].

C. Likelihood formulation and Optimization Strategies

In general, we do not have access to the exact pair-wise metric values and can only observe a noisy and distorted version of them, usually obtained by actively probing the network. If we have a statistical model that relates the underlying (unknown) metric values and the measurements, we can formulate the topology identification problem as a maximum-likelihood estimation exercise.

1) Bottom-up agglomerative procedure.

In a scenario where one can determine the true pair-wise similarity metrics γ, it is possible to reconstruct the tree topology using a simple agglomerative bottom-up procedure. When we only have access to the measurements x, conveying indirect information about γ, we can still develop a bottom-up agglomerative clustering algorithm to estimate the true topology. This method follows the same conceptual framework as many hierarchical clustering techniques, and proceeds by repeatedly applying four steps:

1. Choose the pair of nodes with the highest similarity.
2. Merge the pair into a new node/cluster.
3. Update the similarities between the new node and the former existing nodes.
4. Repeat the procedure until only one node is left.

The crucial step is the update of the similarity values, and in many hierarchical clustering algorithms the update procedure is chosen via application-dependent heuristics [10]**.** In our model-based approach, which relates γ to x, the appropriate update of similarities arises naturally from the likelihood formulation and leads to the agglomerative likelihood tree (ALT) algorithm. The algorithm commences by considering a set of nodes S, initialized to the receiver set *R*, and forming the estimates of the pair-wise similarity metrics for each pair of nodes in the set S, given by

$$\gamma_{ij} = \arg\max\left(f_{ij}\left(x_{ij} \mid \gamma\right) + f\left(x_{ji} \mid \gamma\right)\right) \quad \text{for } i, j \in S,\ i = j \qquad (7)$$

One expects the above estimated pair-wise similarities to be reasonably close to the true similarities γ**.** Consider the pair of nodes such that γ_{ij} is greatest, that is, $\gamma_{ij} >= \gamma_{lm}$, for all $l,m \in$ S. We infer that *i* and *j* are the most similar nodes, implying that they have a common parent *k* in the tree. Assuming that our decision is correct, the tree structure and the likelihood impose some structure on the true similarities, providing a logical way to perform the merging of similarities. The algorithm proceeds by replacing nodes *i* and *j* with their parent *k* in S. For a given node *k*, we denote by R_k the set of receivers which are descendants of *k* in the tree. Thus, at the initial stage of the algorithm $R_i = \{i\}$, and after the update step, $R_k = R_i \cup R_j$.

We update the similarity estimates in S according to

$$\gamma_{kl} = \gamma_{lk} \equiv \arg\max \sum_{\gamma \in R} f_{rl}\left(x_{rl} \mid \gamma\right) + f_{lr}\left(x_{lr} \mid \gamma\right) \quad \text{where } l \in S \setminus \{k\} \qquad (8)$$

These two steps, selecting the pair of nodes with maximum estimated similarity for merger and updating the similarities, are repeated until there is a single node in S. [5] formalized the concepts behind this algorithm and showed that if the underlying tree is binary and the estimated pair-wise similarities are sufficiently close to the true similarities, then

the ALT algorithm is equivalent to the MLT and identifies the true topology.

2) Markov chain Monte Carlo approach.

Despite the simplicity of the ALT algorithm, it is a greedy procedure based on local decisions that involve the estimated pair-wise similarities. If an incorrect local decision is made at some stage in the algorithm, then it cannot be reversed. In the topology estimation problem the measurement process is generally distributed, relying on clocks and counters at numerous network sites. It is frequently the case that several of the measurements are substantially more inaccurate than the rest. The ALT algorithm compares pair-wise similarity estimates, each of which is formed from only a subset of the available measurements and is thus vulnerable to the effect of the local inaccuracies. Unlike the ALT, the MLT estimator takes a global approach: the expression to be optimized in (8) involves a contribution from all of the measurements, and identification of the MLT requires a simultaneous consideration of all the pair-wise similarities. The price to pay is that identification of the MLT involves a search over the entire forest F. In this section we propose a random search technique that efficiently searches the forest of trees and, most importantly, focuses on the likely regions of the forest. For a given set of measurements x we can regard the profile likelihood $L(x|T)$ as a discrete distribution over the set of possible tree topologies F (up to a normalizing factor). One way to search the set F is to sample it according to this distribution. The more likely trees are sampled more often than the less likely trees, making the search more efficient. The sampling can be implemented using the Metropolis–Hastings algorithm [11]. For this we need to construct a Markov chain with state space F. We allow only certain transitions. For a given state (a tree) $si \in F$ we can move to another state (tree) using "birth moves" and "death moves" as illustrated in Fig. 3. Details of the entire procedure can be found in [5]. The Metropolis–Hastings algorithm is a basic sampling approach, which, despite its simplicity, results in improved performance compared to ALT; the incorporation of more sophisticated sampling strategies is an avenue for developing improved topology identification procedures. To achieve our (approximate) solution of (8), we simulate the constructed chain and keep track of the tree we visit that has the largest likelihood; the longer the chain is simulated, the higher the chance of visiting the MLT at least once.

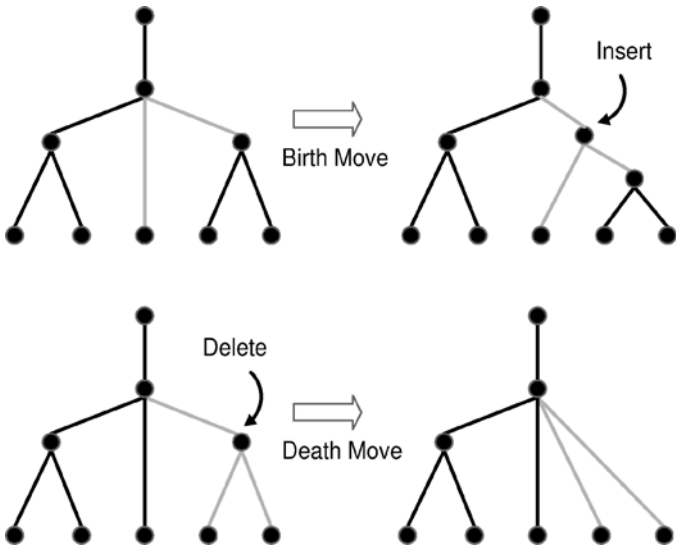

Fig. 3. Illustration of the birth and death moves in the MCMC search algorithm. The birth move selects a node with more than two children chooses two of these children and inserts an extra node as the new parent of these children. The death move chooses a node with two children and deletes that node.

Although theoretically the starting point (initial state) of the chain is not important, provided that the chain is simulated for long enough, starting at a reasonable point improves the chance of visiting the MLT in a reasonable simulation period. Starting the chain simulation from the tree obtained using the ALT algorithm is a reasonable approach, since this is a consistent estimator and so one expects the resulting tree to be "close" (in terms of the number of MCMC moves) to the actual MLT. This is the major reason the simple Metropolis–Hastings sampling procedure works reasonably well. Although inefficiencies can prevent it from visiting more than a small region of the forest, it does visit much of the region near the MLT early in its evolution and can thus "correct" local errors in the ALT.

One drawback to the likelihood criterion is that it places no penalty on the number of links in the tree. As a consequence, trees with more links can have higher likelihood values (since the extra degrees of freedom they possess allow them to fit the data more closely). This is an instance of the classic "over-fitting" problem associated with model estimation [12] and can be remedied by applying regularization, that is, by replacing the simple likelihood criterion with a *penalized* likelihood criterion,

$$T_\lambda = \arg\max_{T \in F} \log \ell(x \mid T) - \lambda N(T) \qquad (9)$$

where $n(T)$ is the number of links in the tree T and $\lambda \geq 0$ is a parameter, chosen by the user, to balance the trade-off between fitting to the data and controlling the number of links in the tree. We can use an MCMC method in a similar fashion as before to approximately find the solution of (9). Minimum description length principles [12] motivate a penalty that is dependent on the size of the network (in terms of the number

of receivers). However, other model selection techniques lead to choices of different penalties [13].

Deploying measurement probing schemes and evaluating inference algorithms for larger networks is the next key step in network monitoring. The impact of network monitoring, on network control and provisioning could become the application area of most practical importance. Admission control, flow control, service level verification, service discovery and efficient routing could all benefit from up-to-date and reliable information about link and router level performances.

III. Acknowledgment

The authors gratefully acknowledge the contributions of Rui Castro, Mark Coates, Gang Liang, Robert Nowak and Bin Yu for their work on the original version of this document

IV. References

Periodicals:

[1] CÁCERES, R., DUFFIELD, N., HOROWITZ, J. and TOWSLEY, D.(1999). Multicast-based inference of network-internal loss characteristics. IEEE Trans. Inform. Theory pp. 2462–2480.

[2] CAO, J., DAVIS, D.,VANDER WIEL, S. and YU, B. (2000a). Time varying network tomography: Router link data. J. Amer. Statistics. Assoc. pp. 1063–1075.

[3] LO PRESTI, F., DUFFIELD, N., HOROWITZ (2002). Multicast-based inference of network-internal delay distributions. IEEE/ACM Transactions on Networking pp. 761–775.

[4] LO PRESTI, F., DUFFIELD, N., HOROWITZ, J. and TOWSLEY, D. (2002). Multicast-based inference of network-internal delay distributions. IEEE/ACM Transactions on Networking pp. 761–775.

Books:

[5] COATES, M., CASTRO, R., NOWAK, R., GADHIOK, M.KING, R. and TSANG, Y. (2002a). Maximum likelihood network topology identification from edge-based uni-cast measurements. In *Proc. ACM SIGMETRICS 2002* 11–20. ACM Press, New York.

[6] BESTAVROS, A., BYERS, J. and HARFOUSH, K. (2002). Inference and labeling of metric-induced network topologies. In *Proc. IEEE INFOCOM 2002* 2 628–637. IEEE Press, New York.

[7] WILLET, P. (1988). Recent trends in hierarchical document clustering: A critical review. *Information Processing and Management* **24** 577–597.

[8] HASTINGS, W. (1970). Monte Carlo sampling methods using Markov chains and their applications. Biometrika pp 97–109.

[9] MORRIS, R. and LIN, D. (2000). Variance of aggregated web traffic. In Proc. IEEE INFOCOM 2000 pp 360–366. IEEE Press,New York.

[10] IHAKA, R. and GENTLEMAN, R. (1996). R: A language for data analysis and graphics. J. Comput. Graph. Statist. pp 299–314.

[11] KELLY, F. P., ZACHARY, S. and ZIEDINS, I., eds. (1996). *Stochastic Networks*: *Theory and Applications*. Oxford Univ. Press.

[12] LO PRESTI, F., DUFFIELD, N., HOROWITZ, J. and TOWSLEY, D. (2002). Multicast-based inference of network-internal delay distributions. *IEEE/ACM Transactions on Networking pp* 761–775.

[13] RATNASAMY, S. and MCCANNE, S. (1999). Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *Proc. IEEE INFOCOM 1999* **1** 353–360. IEEE Press, New York.

Technical Reports:

[14] CAO, J., VANDER WIEL, S., YU, B. and ZHU, Z. (2000b). A scalable method for estimating network traffic matrices. Technical report, Bell Labs.

V. Biographies

Devashish Patel is a student of Fourth year Information Technology Engg. at S.P. College of Engg, Mumbai.He has attended the All India IEEE Students Congress .His areas of interest include Artificial Intelligence and Thoretical Physics, core systems programming

(devashish_p@yahoo.com)

Rohan Rambhia is a student of Fourth Year Information Technology Engg at S.P.College of Engg, Mumbai. He is a High School Scholarship recipient.. His Areas of interest includes applied mathematics, software designing and core application and systems programming.

(rohan.rambhia@yahoo.com)

Amit Rander is a student of Fourth Year Information Technology Engg at S. P. College of Engineering, Mumbai. He is a recipient of Maharashtra State talent Search scholarship. His area of interest includes DSP, Image Processing and Multimedia Commerce, Web technology.

(amitrander@yahoo.com)

Kailas K Devadkar is Lecturer in Information Technology Dept, S. P. Institute of Technology, Mumbai. His areas of interests include database and computer networks

(kailaskd@rediffmail.com)