Flow Generation for IP/ATM Label-Switched Routing over Random Networks

Aaron Harwood and Hong Shen
School of Computing and Information Technology
Griffith University
Nathan, QLD 4111, AUSTRALIA
Email: {A.Harwood, H.Shen}@cit.gu.edu.au

Abstract

We address the problem of generating ATM labels which facilitates IP packet flow through the network. We define the virtual flow topology and provide a stochastic algorithm, $GFLOW$, that generates labels for virtual connections using periodic broadcasts providing simple and efficient robustness and oblivious execution. For a random network with N nodes of average degree $\bar{d}$ and dimeter $\Theta(k)$, we demonstrate how our algorithm can be used to generate a mean $l = 1 + (k-1)\bar{d}$ labels at each node to provide a probability $\Theta\left(\frac{l}{N}\right)$ that any pair of nodes will have a virtual connection between them. We show that with probability roughly $\frac{1}{2} + \frac{l}{2N}$ any node may route a message along a virtual connection which terminates within an ϵ-neighborhood of the destination, where $\epsilon = \Theta\left(\log_{\bar{d}}\left(\frac{N}{k}\right)\right)$, with l as stipulated. Of course the number of labels generated at each node is variable and directly relates to the cost in such a way that a network administrator can trade label space for increased performance. We provide simulation results using Matlab mathematical language interpreter that supports our analysis.

1 Introduction

The combination of ATM label switched routing with IP packet forwarding is a rapidly growing technology[7, 5]. A fundamental problem is to generate ATM labels which facilitates IP packet flow through the network[6]. The use of ATM virtual circuit switching under IP provides, for flow labeled incoming packets, direct routing to the output port, known as label-switched routing (LSR) where packets are labeled according to their flow using labels locally identified between neighboring routers. Labels can be assigned to aggregate flows (flows with similar destination networks) and according to traffic patterns in order to reduce the total labels required over a network[4]. Just as important though, the algorithm should be robust, easy to implement and executable obliviously at each router, akin to the successful features of IP packet forwarding algorithms widely used for the Internet.

1.1 Previous Work

The λ *Flow Problem* as defined in [3] is to provide a flow topology such that the maximum number of flows any packet must traverse to reach a destination is no greater than λ and the maximum table size over all routers is minimized. This problem is identical to the open problem of minimizing the cost of an interconnection network as was demonstrated by us earlier. In [2] Faragó and others develop techniques to construct optimal virtual path network topologies using random graphs. There algorithms are off-line and don't relate directly to routing protocols, however their results like other more general results[1] indicate that random constructions of networks, such as the one presented in this paper, do indeed exhibit optimal characteristics. Other work that we know of also invariably requires centralized processing of some sort in order to construct the virtual topology or constructs a topology in response to traffic patterns without regard for the topology.

For example, recent work by others in *INFOCOM 2000*[1] also addresses this issue. Afek and Bremler-Barr propose a scheme drawing analogies to a subway system in a large metropolitan area. Packets are put onto trains (flows) to be removed from the train after some number of stations (routers). Thus, their scheme requires a counter and extra hardware functionality at Layer 2. They look at using multi-train lines (several flows) but leave optimization of the number of labels versus the number of flows for future research. Furthermore their algorithms require constructing topological maps. For n routers they reduce the table size

[1] http://www.cse.ucsc.edu/ rom/infocom2000/

(labels recorded at each node) to $\sqrt{n}$.

1.2 Our Contribution

In this paper we define the virtual flow topology and propose a stochastic algorithm, $GFLOW$, that generates labels for virtual connections using periodic broadcasts to provide simple and efficient robustness and oblivious execution. Our proposed allows the mean table size to be set according to a given function, needs no topological maps and extra functionality at Layer 2, and can be easily implemented.

2 Flow Topology Definition

Let the network be $G(V, E)$ and have distance from u to v over E being $\delta(u, v)$ traversals, where $u, v \in V$, $E \subseteq V \times V$. A flow consists of a destination node, v, plus one or more nodes along a path, $F = \{v, v_{-1}, v_{-2}, \cdots, v_{-j}\}$. We use negative subscripts to indicate *upstream* nodes, ie flows are directed and v_i is upstream from v_j if $i < j$. Conversely we may identify a flow from the flow "leaf" or last upstream node $u = v_{-j}$, so that $F = \{u, u_1, u_2, \cdots, u_j\}$. Here positive numbers mean *downstream*. Each node on the flow must record a label which it associates with the destination and a mapping from that label to an output port (neighbor) with possibly a new label to switch to. Labels have significance only between neighboring nodes.
A *flow tree on* v is the set of flows that terminate at v. A flow path from u to v consists of 2 or more flows, $F_1, F_2, \cdots, F_j$, where the destination of F_i must be equal to the leaf of flow F_{i+1}. Of course the leaf of F_1 must be u and the destination of F_j must be v.

The set of flows $\mathcal{F}$ generated over G is the *flow topology*. The graph obtained as that induced on G by $\mathcal{F}$ has vertices consisting of the leaf and destination nodes of each flow in $\mathcal{F}$ and edge set being connections between the leaf and destination of each flow. The degree of a node is equivalently the number of labels, with l_u being the number of labels recorded by node u and l the maximum over all nodes. In this way we can assess the asymptotic cost of the virtual topology.

3 On the Growth of Neighborhoods over Random Networks

Consider a node $v \in V$ and neighborhood network $G_v(\epsilon)$ such that $u \in G_v$ iff $\delta(v, u) \leq \epsilon$. Let $\Pi_v(\epsilon)$ be the boundary network of $G_v(\epsilon)$ such that $u \in \Pi_v(\epsilon)$ iff $\delta(v, u) = \epsilon$. Assume n_ϵ is the number of nodes in $\Pi_v(\epsilon)$ and that the function n_ϵ is independent of the node v. In fact, this assumption is justified if the number of edges contained in the network is large enough, as we know that, with sufficient edges, a random network will contain with high probability a single large connected component (with some small isolated components that are negligible) of which there are many cycles of various lengths. Given that these cycles are uniformally distributed over the component, the neighborhoods of different nodes will, with all likelihood, furnish statistically equivalent structures.
The proof of our assumption lies in direct inference from the work of Erdös and Rényi, who show that the evolution of random graphs takes on five distinct phases classified by the probability of the random graph exhibiting certain structures[8].

We have occasion to ask for an expression of n_i in terms of the average degree, $\bar{d}$, and the total number of nodes, N. In these terms it is reasonable for instance that

$$n_0 = 1;\ n_1 = \bar{d}$$

It is less clear the value for n_2. To proceed let us examine the basic random process of picking a number of edges $M = \bar{d}N/2$, each with equal probability, ie, after the first edge is picked, of the remaining edges, each edge will be picked with equal probability. Furthermore examine the case when partitioning the N nodes into two sets, one set, $G_v(i)$, containing N_i nodes and the other set, $G - G_v(i)$, containing $N - N_i$ nodes. One may ask for the probability that an edge will be chosen at random such that the choice satisfies one of three mutually exclusive possibilities: both vertices of the edge are inside, outside, and one inside and the other outside of $G_v(i)$.

Enumerating the possibilities there are $N_i(N - N_i)$ total possible edges that could cross the boundary an odd number of times, $C^2_{N-N_i} = \binom{N-N_i}{2}$ possible edges that could be outside $G_v(i)$ and $C^2_{N_i} = \binom{N_i}{2}$ edges that could be inside $G_v(i)$. The total number of edges is

$$C^2_N = \binom{N}{2} = \frac{N(N-1)}{2}.$$

After discretely selecting M edges, and allowing T to be the random variable giving the number of edges that cross the boundary, the probability that exactly $T = t$ of them cross the boundary is

$$P[T = t] = \binom{N_i(N - N_i)}{t} \binom{\binom{N-N_i}{2} + \binom{N_i}{2}}{M - t} \binom{\binom{N}{2}}{M}^{-1}.$$

To confirm this is a probability see that $\sum P[T = t] = 1$, where $t = 0, 1, \cdots, M$ or

$$\sum_{t=0}^{M} C^t_{N_i(N-N_i)} C^{M-t}_R = \binom{\binom{N}{2}}{\frac{\bar{d}N}{2}}$$

where $R = C^2_{N-N_i} + C^2_{N_i}$. This is clearly true when applying the rule $C^k_{m+n} = \sum_{j=0}^{k} C^j_m C^{k-j}_n$ since $N_i(N -$

$N_i) + C^2_{N-N_i} + C^2_{N_i} = C^2_N$. The expected value, $E[T] = \sum_{t=0}^{M} tP[T = t]$, is not nearly as easy to calculate. Using *Chebyshev's summation inequalities* we can but say that

$$E[T] \leq \frac{M-1}{2}. \qquad (1)$$

Allowing for edges to be *continuously* selected, so as to provide unchanging proportions of probable outcomes, we may say that roughly

$$E[t] = \frac{N_i(N-N_i)}{\binom{N}{2}} M = \frac{N_i(N-N_i)}{N-1}\bar{d}$$

which gives for $N_i = N/2$ a value only slightly greater than that deduced in Eq. (1). Assume, that of the remaining $N - N_i$ nodes, each node is of equal probability of being connected to an edge leaving $G_v(i)$. As we are not interested in the exact topology, but only the expected number of nodes, we have only to ensure when counting that no two edges connect the same node pair. For each node in $G_v(i)$ there are $\binom{N-N_i}{\frac{N-N_i}{N-1}\bar{d}}$ ways of connecting the outgoing edges. Each candidate node appears in $\binom{N-N_i-1}{\frac{N-N_i}{N-1}\bar{d}-1}$ combinations which gives purely a probability $\bar{d}/(N-1)$ of being selected by a single node in $G_v(i)$ since $C^k_n = (n/k)C^{k-1}_{n-1}$. This is the intuitive result if we consider that the degree of any node is uniformly distributed over the remaining nodes in a random network. With exactly N_i independent trials, the chance of *not being selected* is

$$\left(1 - \frac{\bar{d}}{N-1}\right)^{N_i}$$

so for $N - N_i$ nodes roughly

$$(N - N_i)\left(1 - \left(1 - \frac{\bar{d}}{N-1}\right)^{N_i}\right)$$

of them will be selected.

We now consider the number of nodes in the band i, n_i, as only those nodes can connect to nodes forming the next band. Note that each of the $N' = N - N_i + n_i + n_{i-1} = N - N_{i-2}$ nodes has probability $1/(N'-1)$ of receiving an edge from a node in band i. With exactly n_i trials, we calculate the fraction of the N' nodes that form band i to get:

$$\begin{aligned} n_{i+1} &= (N - N_i)\left(1 - \left(1 - \frac{\bar{d}}{(N - N_{i-2} - 1)}\right)^{n_i}\right) \\ N_{i+1} &= N_i + n_{i+1} \end{aligned}$$

4 Probabilistic Flows

Many algorithms may be devised that distribute labels throughout a network to provide flows. Recall the number of labels recorded by a node u is l_u and l is the maximum over all nodes. Our first consideration is of course that if node $u \neq v$ knows of v (ie associates a label with it) then a neighbor of u must also know of v (let every node know of itself). Typically this neighbor will be on the shortest path to v.

To illustrate this concept and to provide a framework for analysis consider an intuitive algorithm for generating flows over a general topology. The proposed algorithm works similar to distance-vector routing in that suggested flows that are of length greater than a known flow length are immediately discarded.

Algorithm: **GFLOW**

1. Every node generates a label for itself, ie $P[(u, u) \in T] = 1$.
2. Periodically[2], every node broadcasts its own label to its neighbors.
3. A node u receiving a label, y, associated with v over a minimum path, discards any current knowledge of v and then does exactly one of the following
   (a) Discards the received information (losing all knowledge of v).
   (b) Records y and Broadcasts the received information plus a arbitrary generated pre-image of y, x, to its neighbors.

End: **GFLOW**

When a node broadcasts a flow label to its neighbors it also broadcasts the length of the flow in hops, alike to distance-vectoring so that only the minimum length flows are recorded.

To enforce Eq. (**??**) we may have that a node u cannot discard knowledge of v if an upstream neighbor of u, u_{-1}, has not discarded knowledge of v. If all upstream neighbors of u have discarded knowledge of v (or were never given it from u) then u can choose to discard its knowledge of v at any time. The node u is in this case a "leaf" node of the flow. Otherwise u must inform u_{-1} that it intends to discard knowledge, breaking the flow. It may also simultaneously declare a new knowledge of v and a new label for u_{-1} to switch to. Every upstream node must also discard knowledge of v unless it has redundant alternatives.

Let $GFLOW$ be executed on all nodes in an oblivious way. Also, let the decision as to whether information is Discarded or Recorded and Broadcast be purely a random one with probabilities p_D and p_R respectively. In other words, after receiving a valid label (one that comes from a shortest path route from v) the algorithm chooses one of two possibilities:

[2]The period may be measured in hours or days.

1. Discard the knowledge or
2. Record and Broadcast the knowledge.

Clearly $p_D = 1 - p_R$. We want to calculate the probabilities p_D and p_R that will give an average number of nodes $\mu_v(\epsilon)$ that know of v out of the expected n_ϵ nodes at distance ϵ from v.

The exact nature of this probability depends on many factors including the local neighborhood of u or v, the technology being used, the traffic statistics etc. For now assume that we want

$$P[(u,v) \in T] \propto \frac{1}{\delta(u,v)}.$$

In this way, nodes far from v will have a small probability of knowing of v, while nodes closer to v will have a large probability of knowing of v. We may stipulate for instance, when the algorithm is in equilibrium, what the mean concentration of knowledge will be for any particular node. Using μ_ϵ to be the mean number of nodes at distance ϵ from v that will have knowledge of v and knowing n_ϵ is the expected number of nodes at distance ϵ from v we have

$$P[(u,v) \in T] = \frac{\mu_{\delta(v,u)}}{n_{\delta(v,u)}}.$$

Over a suitable time interval, proportionally longer than the broadcast period of $GFLOW$, every node may be observed to record information at a given probability.

A non-leaf node can discard knowledge of v so long as it instructs all upstream nodes that use that knowledge to also discard knowledge of v. Thus the probability that u discards knowledge of v is 1 minus the product of probabilities that all down stream nodes do not. In any case we must have p_D or p_R as a function of ϵ since $P[(u,v) \in T]$ as given is a function of ϵ. From $GFLOW$ definition:

$$\mu_0 = 1,\ p_R(0) = 1,\ p_R(1) = \mu_1/n_1.$$

Working inductively we assume there are μ_i nodes in band i which in total connect to a fraction $(1-(1-\bar{d}/N')^{\mu_i})$ of the nodes in band $i+1$ where $N' = N - N_{i-2} - 1$. Of the total connected nodes we want only μ_{i+1} of them to record a label which yields the probability

$$p_R(i+1) = \frac{\mu_{i+1}}{(N-N_i)(1-(1-\bar{d}/N')^{\mu_i})}.$$

4.1 Simulation Results

We implemented the above algorithm on networks of various size and average degree. Our simulations though are rather limited to small cases, no larger than 200 nodes. We observed the flow trees constructed using one iteration of the algorithm (ie one broadcast for each node in the network). Since each iteration is independent this does not lead to loss in generality. For every node v in the network we observed the number of labels that were recorded at each distance ϵ from v and averaged this observed function over all the nodes. In other words we calculated $\bar{\mu_\epsilon}$ based on observed results. Fig. 1 and Fig. 2 show the observ ations for the case for a random network of 100 nodes with $\bar{d} = 3$ and a random network of 100 nodes with $\bar{d} = 5$ respectively. Each plot shows five trials of the algorit hm with $\mu_0 = 1$ and $\mu_i = \bar{d}$ for all $i > 0$.

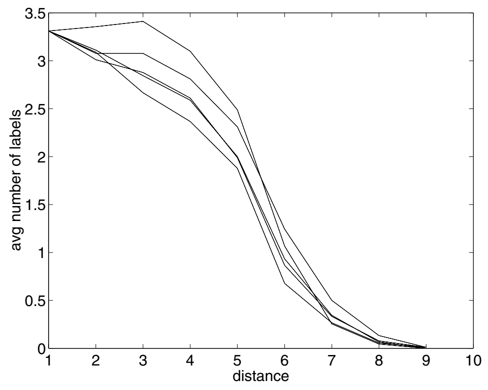

Figure 1. Average labels recorded versus distance.

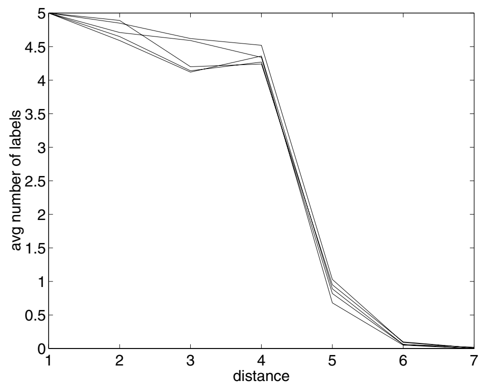

Figure 2. Average labels recorded versus distance.

There are a number of factors which lead to a deviation from the desired values. Firstly, the large deviation at large distance is due to the fact that for instance it is impossible to have 3 nodes record information when the band consists of less than 3 nodes. Also, our simulation required a connecte d network but we extracted the largest connected component from the random network which has a number of nodes slightly less than that quoted and furthermore has statist ical properties that differ slightly from a random network. In general the results in dicate that the algorithm operates as we expect it to, providing for each node a number of possible flows that are distributed over each band in the network. Also they provide an i ndication of the variation which we have not yet touched upon. Variation is a parameter wh ich we leave for future derivation. Fig. 3 shows two trials of the algorithm for a random network of almost 200 nodes and $\bar{d} = 3$.

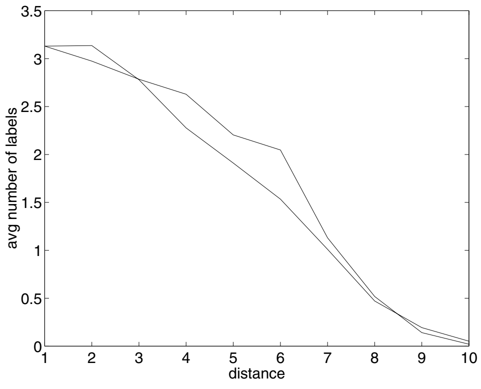

Figure 3. Average labels recorded versus distance.

5 Analysis

The number of labels dedicated to different services is a design criteria that necessarily impacts the performance of that service. To have with probability 1 a flow between every pair of nodes we would need every router to record N labels. But N is considered high cost so the purpose of this section is to show the relationship between the cost of the algorithm in terms of the labels recorded at each node and the estimated number of flows that will be required for any random route request.

To begin, recall that μ_ϵ is the number of nodes at distance ϵ from a node v in the network that on average will record a label for v. Assuming that $P[(u,v) \in T] = P[(v,u) \in T]$ it follows that

$$l = \sum_{i=0}^{k} \mu_i. \tag{2}$$

Of course we want that $l = o(N)$ so as to significantly reduce the cost. Let a route request be generated at node u with a destination node v chosen at random; recall the distance from u to v is $\delta(u,v)$ and $P[(u,v) \in T] = \frac{\mu_{\delta(u,v)}}{n_{\delta(u,v)}}$. If a random request is generated at u then $\delta(u,v)$ is a random variable and

$$E\left[\frac{\mu_{\delta(u,v)}}{n_{\delta(u,v)}}\right] = \sum_{i=0}^{k} \frac{\mu_i}{n_i}\frac{n_i}{N} = \frac{l}{N}$$

is the average probability that u has a label for v. Clearly if $\mu_i = n_i$ then $\bar{P} = 1$ but $l \neq o(N)$ since $\sum n_i = N$. Setting $\mu_i = n_i/q$ obtains $\bar{P} = 1/q$ but still l is not significantly reduced. Let μ_i be a constant equal to for instance $\bar{d}$ (as used in the previous simulations) with $\mu_0 = 1$. From Eq. (2) we have $l = 1 + (k-1)\bar{d}$ which (with respect to k) is directly proportional to the interconnection network cost as defined in the literature so

$$\bar{P} = \frac{1 + (k-1)\bar{d}}{N}.$$

From this result *we conclude that on average, the probability that u can route a message to v using a single flow with $l = O(k\bar{d})$ labels recorded at each node is $\Theta(l/N)$.*

Clearly if u cannot route the message to v using a single flow, then u must route the message to a node as close to v as possible. The default case is that u routes the message to a node one step closer to v. The message in this case will traverse a number, λ, of flows.

Now, every flow tree is independent, so in an average sense it is equally likely that u has recorded a label for a neighbor of v. Thus if u cannot route to v, it may route to a neighbor of v. Using these average probabilities of routing it follows that for any q nodes chosen arbitrarily, u has probability

$$R_q = 1 - \left(1 - \bar{P}\right)^q \approx q\bar{P} - \frac{q(q-1)}{2}\bar{P}^2 \tag{3}$$

of recording a label for at least one of them. Since these nodes may be chosen arbitrarily, choose a set of nodes that form a neighborhood of v such that $q = N_\epsilon$ where recall that $N_\epsilon = \sum_{i=0}^{\epsilon} n_i$. The probability that u may route to a neighborhood of v, or in other words that u may route to a node of distance no greater than ϵ from v is now computable. If we let $q = N/l$ we see that

$$R_q = \frac{1}{2} + \frac{l}{2N}.$$

Now, it is quite clear that for values of $0 < i < \frac{k}{2}$ we can have $n_i \approx \bar{d}^i$. Let us assume for this analysis that the network is infinitely large so that (for all finite i)

$$N_i = \sum_{j=0}^{i} n_j \approx \frac{1 - \bar{d}^{i+1}}{1 - \bar{d}} \tag{4}$$

where setting $N_i = q = N/l$ and solving for i in terms of N, l and $\bar{d}$ yields

$$i = \log_{\bar{d}}\left(1 - \frac{N}{l}(1 - \bar{d})\right) - 1 = \Theta\left(\log_{\bar{d}}\left(\frac{N}{k}\right)\right).$$

In this way we conclude that *a random request has a probability roughly $\frac{1}{2}+\frac{l}{2N}$ of routing to a node within a distance $\Theta\left(\log_{\bar{d}}\left(\frac{N}{k}\right)\right)$ of the destination.* More succinctly we can write the probability that u can route to a node at distance no greater than i from v by substituting q in Eq. (3) with the expression for N_i from Eq. (4):

$$\begin{aligned} R_{N_i} &\approx N_i\bar{P} - \frac{1}{2}(N_i^2 - N_i)\bar{P}^2 \\ &= \Theta\left(\frac{k\bar{d}^{i+1}}{N}\right). \end{aligned}$$

For $i = 0$ we get the kd/N, the same probability calculated for routing directly to the node using a single flow. The probability of routing to an i-neighborhood of the destination increases exponentially in $\bar{d}$ with i.

Calculation of the expected value for λ cannot proceed using the coarse approximations we have made so far. However we can see that λ is bounded by $i+1$ where i has probability distribution given above. In other words $P[\lambda = 1] = kd/N$ and $P[\lambda = 2] \leq kd^2/N$, etc. We obtain this bound by observing that after the first flow we have at most i flows to go. So from our previous result we see that about half of all requests will require to traverse $\Theta\left(\log_{\bar{d}}\left(\frac{N}{k}\right)\right)$ flows.

However, this bound is quite loose as really we understand that after traversing the first flow the message is appreciably closer to the destination. Missing from our theory is the probability function that provides the number of nodes in the set L_x where $s \in L_x$ if $\delta(u,s) + \delta(v,s) = x$. Using this function we could properly derive an estimation for λ but we leave this problem open for future work.

6 Conclusion

We presented as a main contribution of this paper, a simple and efficient algorithm for generating labels in an IP/ATM-LSR network. The algorithm requires no synchronization between nodes other than immediate neighbors. Each node relies on a probability function to determine whether it records a label or discards a label associated with a given flow termination node. The probability function is computed using only the total nodes in the network, N, and the average degree $\bar{d}$, assuming for generality that the network edges are randomly distributed. The method of generating this probability function is crucial for discussion and we present perhaps the most intuitive case, when the probability of a virtual connection existing between any two nodes is inversely proportional to the distance between the nodes. We leave open the possibility of this probability function being proportional to other factors such as technology, topology, application or traffic statistics.

We suggest that this approach may be used to analyze deterministic algorithms using only probabilities of 1 and 0 for instance. In effect, the performance that our algorithm exhibits (as a function of label space available or dedicated) may be seen as a best average case. It is surely the case that better performance should be obtained by algorithms that use given or inferred knowledge of the topology beyond the simple variables we make use of, *viz.* N and $\bar{d}$. However the strong acceptance of protocols such as IP has been attributed to the fact that little topological knowledge is recorded and/or relied on. Our algorithm thus promises a robust and extensible solution to achieving good performance over general topologies.

References

[1] B. Bollobas. *Random Graphs*. Academic Press, London, 1985.

[2] Farago, Chlamtac, and Basagni. Virtual path network topology optimization using random graphs. In *INFOCOM: The Conference on Computer Communications, joint conference of the IEEE Computer and Communications Societies*, 1999.

[3] Aaron Harwood and Hong Shen. Near optimal flow labelling in ATM/IP-LSR networks using multi-segment flows. Proceedings of IEEE International Conference on Networks, Singapore, 2000.

[4] Ken ichi Nagami, Hiroshi Esaki, Yasuhiro Katsube, and Osamu Nakamura. Flow aggregated, traffic driven label mapping in label-switching networks. *IEEE j. on sel. areas in commun.*, 17(6):1170–1177, June 1999.

[5] Berry Kercheval. *TCP/IP over ATM: a no-nonsense internetworking guide*. P T R Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1998.

[6] P. Newman, T. Lyon, and G. Minshall. Flow labeled ip: A connectionless approach to ATM. In *Proc. IEEE Infocom*, pages 1251–1260, March 1996.

[7] Peter Newman, Greg Minshall, and Thomas L. Lyon. IP switching — ATM under IP. *IEEE/ACM Transactions on Networking*, 6(2):117–129, April 1998.

[8] Joel H. Spencer, editor. *Paul Erdös: the art of counting : Selected writings*. MIT Press, 1973.