Optimizing IP Address Assignment and Static Routing at Large Scale

Jonathon Duerig
University of Utah
duerig@flux.utah.edu

Robert Ricci
University of Utah
ricci@cs.utah.edu

John Byers
Boston University
byers@cs.bu.edu

Jay Lepreau
University of Utah
lepreau@cs.utah.edu

ABSTRACT

We consider the problem of optimizing the assignment of IP addresses to nodes in a network. A good assignment takes into account the natural hierarchy present in the network and assigns addresses in such a way as to minimize the sizes of routing tables on the nodes. Optimized IP address assignment benefits simulators and emulators, where scale precludes manual assignment, large routing tables can limit network size, and realism can matter. It benefits enterprise networks, where large routing tables can overburden the legacy routers frequently found in such networks.

We outline one of the algorithms we are exploring, and describe a key underpinning: a metric, based on Routing Equivalent Sets (RES), that quantifies the extent to which routes to sets of destinations can be aggregated. We present preliminary results of using RES to find assignments that result in small routing tables. When applied to real Internet topologies, we find that our assignment algorithm yields a compression rate of two to three over random assignment.

1. INTRODUCTION

A basic problem in networking is minimizing the size of routing tables on network nodes (hosts and routers). Assuming static shortest-path routes, each node must maintain a routing table that correctly specifies the first hop to each possible destination. Without route aggregation, each of the n nodes in a network must store first hops for all $n-1$ routes. In practice, however, the number of routing table entries can be reduced greatly. For example, with CIDR routing, the routing scheme used in the Internet today, a route for an entire IP prefix can be specified with a single table entry. With the longest prefix matching rule, CIDR addressing easily allows alternative routes to be specified for more specific prefixes.

This work was supported in part by NSF grants CNS–0335296 and CNS-0205702.

Our research seeks to produce a global IP address assignment automatically, i.e., given a network graph $G = (V, E)$, we consider the problem of *address assignment* in which IP addresses must be assigned to each vertex in the graph. We focus on the metric of *routing table size*, which we define to be the number of entries (CIDR prefixes) in a given routing table. Our objective is to develop methods for address assignment that minimizes the average routing table size across all nodes in the network.

Solving this address assignment problem can be important in emulation and simulation environments where test topologies are usually not annotated with IP addresses. Not only is this a difficult problem for a user or operator to solve manually, but it can also matter to solve it well, as the space consumption for routing tables in simulation is significant. Large routing tables have the same negative impact on performance in these environments as they do in reality.

Two directions associated with CIDR addressing influence our work. First, a number of guidelines have been proposed to facilitate manual assignment of IP addresses [2, 3] to make it easier for humans to take advantage of CIDR routing. Second, a method is now known which minimizes the number of table entries for a *given* set of routes and IP addresses. The Optimal Routing Table Construction (ORTC) [1] technique optimally compresses IP routing tables using CIDR rules. ORTC takes a routing table as input and produces a compressed routing table with the same functional behavior, taking advantage of CIDR rules to aggregate routes where possible. For any IP address assignment, ORTC optimally minimizes the number of routes in the routing table. We employ ORTC as a post-processing step in our work.

2. OUR METHODS

The task of assigning IP addresses to minimize routing table size for general graphs is NP-complete, making exact solutions for large graphs intractable. We employ a combination of preprocessing techniques and heuristic methods to generate good address assignments.

For preprocessing, we can optimally assign addresses in polynomial time to certain network structures, such as trees. Furthermore, we have methods of partitioning the network graph into subgraphs which can be named independently, reducing the size of the problem. Since some of these subgraphs are often trees, those portions of the graph can be optimally assigned. After preprocessing, we typically have a richly connected core for which we must compute an IP assignment.

One of the promising heuristics we are investigating is a bottom-up greedy tournament algorithm that we describe in more detail in the next section. We have been investigating a number of alternate heuristics as well. First, a top-down approximation is a plausible replacement for our current bottom-up approximation. This can be done by recursively partitioning the graph. METIS [4] is a well-known graph partitioning package that has worked quite well in our experiments. Second, we have tried mapping the graph onto an IP tree in two disjoint steps: 1) mapping the graph to an ordering and 2) mapping the ordering to a tree. For the first step, we have considered both spectral and combinatorial methods, and both appear promising. Mapping an ordering to a tree is a less well studied problem, for which we are exploring some potential heuristics. A comparative evaluation of all of these methods will be described in detail in an upcoming paper.

2.1 Tournament Heuristic

One of our more promising heuristics uses a greedy tournament strategy to cluster similar nodes and to build a binary tree mapping to the IP topology. Initially, there are n singleton groups, one per node. In each of $n-1$ rounds, we select the pair of groups that when combined, yields the highest score. Then we merge the two selected groups into a single group. After all $n-1$ rounds, the result is a binary tree structure which can be trivially converted to IP addresses.

The scoring function described in the next section can be computed in linear time. When two groups are combined only a linear number of scores need to be recalculated. Since there are $n-1$ rounds, the time complexity of this algorithm is $O(n^3)$. Scores must be stored for each pair of groups and thus the space complexity is $O(n^2)$.

2.2 Routing Equivalence Sets

We score the tournament using Routing Equivalence Sets (RES). The routing equivalence set of a destination set is the set of source vertices whose first hop is the same to every vertex in the destination set. The destination sets are the groups in the tournament heuristic described above.

More formally, let V be the set of vertices in a graph. Let D be a destination set. Let $H_x[y]$ be the first hop from source vertex x to destination vertex y. We define res(D) as:

$$\mathrm{res}\,(D) = \{v \in V : \forall d, e \in D, H_v[d] = H_v[e]\}$$

Naively calculating a RES set from this definition is expensive. When $|D| = n$, the number of vertices in the graph, the time complexity is $O(n^3)$. However, the tournament creates each destination set from the union of two smaller destination sets. If we know the RES of two arbitrary destination sets, calculating the RES of the union of those two sets is linear. First note that by transitivity, for any v and for all $a, b, c \in V$:

$$(H_v[a] = H_v[b] \wedge H_v[b] = H_v[c]) \rightarrow (H_v[a] = H_v[c])$$

The definition of RES and transitivity imply that for destination set D, and specializing to $v \in \mathrm{res}\,(D)$ and $d, e \in D$,

$$H_v[a] = H_v[d] \rightarrow H_v[a] = H_v[e]$$

Which means that $\forall v \in V, \forall d \in D$:

$$\mathrm{res}\,(D \cup \{v\}) = \mathrm{res}\,(D) \cap \mathrm{res}\,(\{v, d\})$$

Therefore, given two destination sets D and E, we can select *any* $d \in D$ and $e \in E$ to give the recurrence:

$$\mathrm{res}\,(D \cup E) = \mathrm{res}\,(D) \cap \mathrm{res}\,(E) \cap \mathrm{res}\,(\{d, e\}).$$

The computation of the RES sets of a single pair can be done in linear time, thus making the calculation of RES for the union of two arbitrary destination sets linear. The greater the RES of a particular destination set, the more source vertices see every destination in that set as equivalent. Therefore the score of a destination set D is $|\,\mathrm{res}\,(D)|$.

	EBONE	Tiscali
No Compression	320,000	530,000
Random with ORTC	28,828	35,260
RES Tournament with ORTC	11,359	16,155
Manual Assignment with ORTC	8,518	8,492

Table 1: Sum of routing table sizes

3. PRELIMINARY RESULTS

The results in Table 1 are based on IP address assignment of two Rocketfuel topologies [5]. EBONE is a topology with 295 nodes and 543 links. The Tiscali topology has 411 nodes and 653 links. Each route count is the sum of the number of routes in the routing tables of all nodes. In these experiments, no preprocessing steps were performed.

The 'No Compression' line is an estimate of the total number of routes in the topology without any routing table compression. This represents the most naive kind of routing table. If the number of nodes is n, and the number of interfaces is i, then the total routing table size without compression is about $n \cdot i$. In 'Random with ORTC' the addresses are assigned randomly, constrained only by the rule that all the interfaces on a LAN have a unique subnet. This illustrates how well ORTC compresses routes without help from good IP address assignment. 'RES Tournament with ORTC' shows the results for the RES tournament described above. We obtain large gains by assigning IP addresses more carefully. The runtimes for this algorithm on a 3.0 GHz desktop PC were 1.2 and 2.5 minutes; we expect preprocessing to have a large impact on performance, especially on large networks. Finally, Rocketfuel graphs provide both a topology and an IP address assignment for that topology. The quality of this assignment, denoted 'Manual Assignment with ORTC,' illustrates how well manual IP assignment can do.

Although our methods perform much better than random assignment, there is still room for improvement.

4. REFERENCES

[1] R. Draves, C. King, S. Venkatachary, and B. Zill. Constructing Optimal IP Routing Tables. In *Proceedings of IEEE INFOCOM*, 1999.

[2] E. Gerich. RFC 1466: Guidelines for Management of IP Address Space, May 1993.

[3] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, and J. Postel. RFC 2050: Internet Registry IP Allocation Guidelines, Nov. 1996.

[4] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. on Scientific Computing*, 20(1):359–392, 1999.

[5] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP Topologies with Rocketfuel. In *SIGCOMM 2002*.