# A CIDR Prefix Stopping Rule for Topology Discovery

Benoit Donnet and Timur Friedman

*Univeristé Pierre & Marie Curie, Laboratoire LiP6-CNRS*
*8, rue du Capitaine Scott*
*75015 Paris - France*

Recently, a first step towards a highly distributed IP-level topology discovery tool has been made with the introduction of the Doubletree algorithm. Doubletree is an efficient cooperative algorithm that allows the discovery of a large portion of nodes and links in the network while strongly reducing probing redundancy on nodes and destinations as well as the amount of probes sent. In this paper, we propose to reduce more strongly the load on destinations and, more essentially, the communication cost required for the cooperation by introducing a probing stop rule based on CIDR address prefixes.

**Keywords:** metrology, topology, traceroute, scalability, prefix

## 1 Introduction

This is a time when highly distributed applications are in full expansion. Among others, we can cite *SETI@home* [ACK+02] (probably the first one and the most famous) and *FOLDING@home* [LSSP02].

The network measurement community is not an exception to this fashion. Some measurement tools have already been released as daemons or screen savers. For instance, recently, we saw the introduction of *NETI@home* [SR04], an application collecting network performance statistics from end-systems.

Tools allowing for topology discovery at the IP level, based on *traceroute* [Jac89], are becoming more distributed. There is a number of well known systems, such as *skitter* [HPMC02], *Ripe NCC TTM* [GGK+01] or *NLANR AMP* [MBB00]. However, the need to increase the number of traceroute sources (the monitors) in order to obtain more complete topology measurement is felt [CM, LBCX03].

The idea of placing a tracerouting tool inside a screen saver, an idea first suggested by Jörg Nonnenmacher as reported by Cheswick et al. in [CBB00] should allow one to quickly obtain a structure of a considerable size. A publicly downloadable measurement tool, DIMES [Y. ct], has been released in September 2004.

Such a large structure has, however, inherent scaling problems. For example, if all the monitors trace towards the same destination, it could easily appear as a distributed denial of service (DDoS) attack. Furthermore, such a system must avoid consuming undue network resources. However, before the development of the Doubletree algorithm, little consideration had been given to how to perform large-scale topology discovery efficiently and in a network-friendly manner.

Doubletree [DRFC05] is based on the tree-like structure of routes in the internet. Routes leading out from a monitor towards multiple destinations form a tree-like structure rooted at the monitor. Similarly, routes converging towards a destination from a set of monitors form also a tree-like structure, but rooted at the destination.

Doubletree acts to avoid retracing the same routes through these structures. A monitor that applies Doubletree probes hop by hop so long as it encounters previously unknown interfaces. However, once it encounters a known interface, it stops, assuming it has touched an already known part of the tree, and the rest of the path to the root is also known. Backwards and forwards probing are thus used. These two probing schemes make use of stop sets. The first one, used during backwards probing and called the *local stop set*, consists of all interfaces already seen by that monitor. Forwards probing uses the *global stop set*

of (interface, destination) pairs accumulated from all monitors. This global stop set is shared amongst all the monitors, in order to keep track of what was already discovered. A monitor that implements Doubletree starts probing for a destination at some number of hops $h$ from itself. It first probes forwards from $h$ and then, backwards from $h-1$. The initial value $h$ is computed based on a probability $p$ of hitting a destination with the first probe sent $h$ hops towards that destination. For a range of $p$ values, Doubletree is able to reduce measurement load by approximately 70% while maintaining interface and link coverage above 90%.

One issue impeding a large-scale deployment of Doubletree is the communication overhead required by sharing the global stop set amongst monitors. For instance, tracing from only 24 monitors towards just 50,000 destinations with $p = 0.05$ will require roughly 20 megabytes [DFC05] for an uncompressed global stop set. This could lead to unacceptable scalability problems when increasing both number of monitors and number of destinations. To reduce the global stop set size, we investigate a lossy compression method: the *Bloom filters* [Blo70]. We found that using Bloom filters allows to reduce the size by a factor of 17.3 with very little loss in node and link coverage.

In this paper, we propose to replace a stopping rule based on destination addresses with a stopping rule based on the *CIDR address prefixes* [FLYV93] of destinations. The idea is to aggregate the destinations set into subnetworks, i.e. we filter each destination address and associate them to a subnetwork with the use of the CIDR address prefixes. Each monitor will probe all the destinations in each subnetwork. In addition to that, the global stop set will contain (interface, destination_prefix) pairs.

The rest of the paper is organized as follow: in Sec. 2, we present our methodology and our results and we conclude in Sec. 3.

# 2 Doubletree with CIDR

## 2.1 Methodology

Skitter data from the beginning of August 2004 serves as the basis of our work. This data set is composed of traceroutes gathered from 24 monitors scattered around the world. All the monitors share a common destination list of 971,080 IPv4 addresses. Each destination is probed in turn by each monitor. To cycle through the destination list, it takes usually three days. For our studies, in order to reduce computing time and hard disk space to a manageable level, we decided to work on a limited destination subset of 50,000 items randomly chosen amongst the whole set.

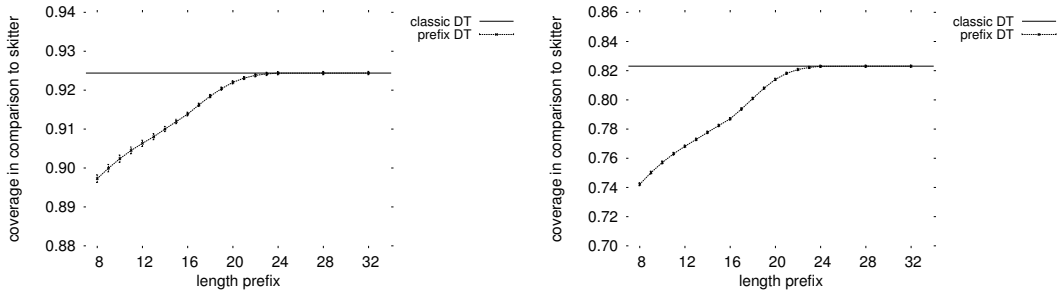
We conduct simulations based on the skitter data, applying Doubletree, as described in [DRFC05]. A single experiment uses traceroutes from all 24 monitors to a common set of 50,000 destinations chosen at random. Each data point represents the average value over fifteen runs of the experiment, each run using a different set of 50,000 destinations. No destination is used more than once over the fifteen runs. We determine 95% confidence intervals for the mean based, since the sample size is relatively small, on the Student $t$ distribution. These intervals are typically, though not in all cases, too tight to appear on the plots.

We use $p = 0.05$, which is a value that belongs to the range of $p$ values that our previous work identified as providing a good compromise between coverage accuracy and redundancy reduction. We test all prefixes length from /8 to /24, as well as lengths /28 and /32 (i.e. full IPv4 addresses). Each monitor probes each destination and records in the global stop set (interface, destination_prefix) pairs instead of (interface, destination) pairs. Compared to classic Doubletree, we only change the global stop set stop rule. Each result considered is compared, in Sec. 2.2, with classic Doubletree.

## 2.2 Results

Fig. 1 shows the main performance metric for a probing system: its coverage of the nodes and links in the network. It illustrates how the nodes and links coverage vary in function of the prefix length. A value of 1.0 (not shown here) would mean that application of Doubletree with the given prefix length had discovered exactly the same set of nodes and links as skitter. As already pointed out in our previous work, the use of Doubletree implies a small accuracy loss in the link and node coverage compared to skitter.

The lowest level of performance is reached for the /8 prefix. In our data set, on average, there are thirteen /8 subnetworks. As these subnetworks are quite large, monitors are stopped early in their probing. The loss




(a) Nodes

(b) Links

**Fig. 1:** Coverage when using prefixes

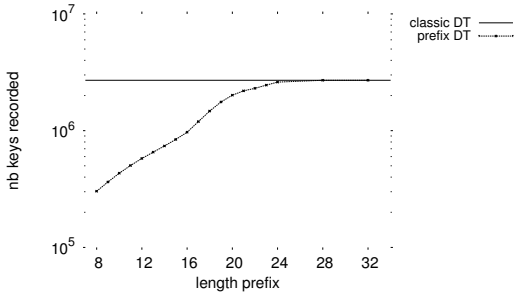



**Fig. 2:** Global stop set size when using prefixes

| | /8 | /16 | /24 | /32 |
|---|---|---|---|---|
| Prefix DT | 2.31 | 7.40 | 19.87 | 20.61 |
| Prefix DT w. BF | 0.361 | 0.689 | 1.192 | 1.192 |
| Classic DT | 20.61 | | | |
| Classic DT w. BF | 1.192 | | | |

**Tab. 1:** Global stop set size comparison (in MB)

of accuracy, however, is not so dramatic. The link coverage is 0,742 instead of 0,823 and node coverage is 0,897 instead of 0,924. We believe that the coverage level is still high due to the way exploration is performed by the first monitor to probe the network. Indeed, this first monitor uses an empty stop set (by definition of Doubletree) and is thus never stopped in its exploration. We further note that performance improves with prefix length until reaching nearly the same accuracy as classic Doubletree with /24 prefixes.

We believe that the loss of accuracy, compared to classic Doubletree, is essentially located within the subnetworks containing destinations but also inside the core of the network, where duplicated links (and the associated nodes) are missed due to the prefix based stopping rule. Typically, probes reach a very few number of destinations in each subnetwork but, in general, they are stopped at the ingress routers. We miss thus essentially the vast majority of destinations located in a given subnetwork. However more nodes and links may be missed if the network structure of the subnetwork is more complex, i.e. the subnetwork is not only composed of an ingress router that connects destinations with the rest of the network.

Fig. 2 shows the number of pairs recorded in the global stop set (in log-scale) as of a function of CIDR block prefixes. We can see that there is a strong reduction for low prefixes. For instance, if we consider a /8 prefix, the global stop set will only contain, in average, 302,854 keys. As each key is recorded as a 64 bit value, it corresponds to a stop set of around 2.31MB. Compared to the classic Doubletree, there is a compression factor of 8.9.

In addition to the mechanism presented in this paper, we could also implement the global stop set as a Bloom filter without losing much coverage accuracy [DFC05, Sec. 3]. Table 1 compared the global stop set implemented as a set of pairs and as a Bloom filter. It also compares classic Doubletree with the mechanism presented in this paper. We see that coupling the prefix based stop rule with a Bloom filter implementation of the global stop set introduces a very strong reduction in the global stop set size. For instance, using a /8 prefix stop rule gives, compared to classic Doubletree, a compression factor of 57.1.

# 3 Conclusion

In this paper, we present an improvement to the Doubletree probing algorithm. By using stop rules based on address prefixes, we show that we are able to strongly reduce communication between monitors while maintaining an acceptable level of coverage accuracy. Further, if we use this simple mechanism with a global stop set implemented as a Bloom filter, we still reduce the global stop set size to very low proportions.

The next prudent step for future work would be to test the algorithms that we describe here on an infrastructure of intermediate size, on the order of hundreds of monitors. We have developed a tool called *traceroute@home* that we plan to deploy in this manner.

# References

[ACK+02] D. P. Anderson, J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. SETI@home: An experiment in public-resource computing. *Communications of the ACM*, 45(11):56–61, Nov. 2002.

[Blo70] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[CBB00] B. Cheswick, H. Burch, and S. Branigan. Mapping and visualizing the internet. In *Proc. of the 2000 USENIX Annual Technical Conference*, San Diego, California, USA, Jun. 2000.

[CM] A. Clauset and C. Moore. Traceroute sampling makes random graphs appear to have power law degree distributions. arXiv:cond-mat/0312674 v3 8 Feb. 2004.

[DFC05] B. Donnet, T. Friedman, and M. Crovella. Improved algorithms for network topology discovery. In *Proc. of Passive and Active Measurement Workshop (PAM)*, Boston, USA, Mar. 2005.

[DRFC05] B. Donnet, P. Raoult, T. Friedman, and M. Crovella. Efficient algorithms for large-scale topology discovery. In *Proc. of ACM SIGMETRICS 2005*, Banff, Canada, Jun. 2005.

[FLYV93] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless inter-domain routing (CIDR): an address assignment and aggregation strategy. RFC 1519, Internet Engineering Task Force, Sept. 1993.

[GGK+01] F. Georgatos, F. Gruber, D. Karrenberg, M. Santcroos, A. Susanj, H. Uijterwaal, and R. Wilhelm. Providing active measurements as a regular service for ISPs. In *Proc. of PAM*, 2001.

[HPMC02] B. Huffaker, D. Plummer, D. Moore, and k Claffy. Topology discovery by active probing. In *Symposium on Applications and the Internet*, Nara City, Japan, Jan. 2002.

[Jac89] V. Jacobsen. traceroute, 1989.

[LBCX03] A. Lakhina, J. Byers, M. Crovella, and P. Xie. Sampling biases in IP topology measurements. In *Proc. of IEEE Infocom '03*, 2003.

[LSSP02] S. M. Larson, C. D. Snow, M. Shirts, and V. S. Pande. FOLDING@home and GENOME@home: Using distributed computing to tackle previously intractable problems in computational biology. In *Computational Genomics*, 2002.

[MBB00] A. McGregor, H.-W. Braun, and J. Brown. The NLANR network analysis infrastructure. *IEEE Communications Magazine*, 38(5):122–128, may 2000.

[SR04] C. R. Simpson, Jr. and G. F. Riley. NETI@home: A distributed approach to collecting end-to-end network performance measurements. In *Proc. of PAM*, 2004.

[Y. ct] Y. Shavitt et al. DIMES, ongoing project.