ComputerScience

Boston University

Interface disambiguation: a non-probe statistical methodology

Larissa Spinelli, Mark Crovella, Brian Ericksson
E-mails:{lspinell,crovella, eriksson}@cs.bu.edu

Abstract

Traceroute probing is a well-known method for discovering links between interfaces on Internet routers. When traceroute is used to produce Internet maps, it is necessary to identify interfaces belonging to the same router — this is called **interface disambiguation**. However, most of the techniques available nowadays use probe methods to perform interface disambiguation and consequently they cannot be applied to historical traceroute datasets. We explore whether one can disambiguate interfaces using characteristics extracted purely from traceroute datasets themselves. The features we consider include topological and addressing properties. We combine these in a statistical fashion. Our results indicate that it is possible to perform highly accurate IP alias resolution without probing.

Motivation

- **Traceroute**: a → u: **a, b, r, t, u**
  a → p: **a, c, k, m, o, p**
  a → l: **a, c, e, f, h, j, l**

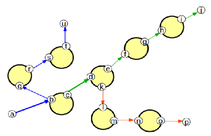

FIGURE 1: Router Graph

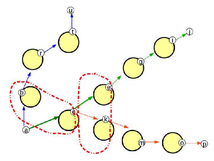

FIGURE 2: Traceroute Graph

- Identify IP addresses belonging to the same router

Disambiguation Techniques

- **Fingerprint:** Send probe packets to different addresses and use response similarities to identify alias
- **Analytical:** Identify alias by analyzing the IP address graph

Analytical Techniques

- **Subnet alignment:** IP addresses in the same subnet are likely to be aliases
- **Hop Count:** IP addresses with a larger percent of common observed hop count elements are likely to be aliases
- **Percent of the common interface in-degree(out-degree):** Interfaces that share many of same incoming(outgoing) interfaces are likely to be aliases

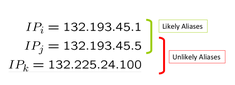

FIGURE 3: Subnet alignment

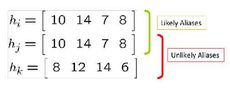

FIGURE 4: Hop Count

FIGURE 5: In-degree

Interface Disambiguation Methodology

- Characteristics: hop count, subnet, in-degree, out-degree
- Use Naive Bayes Rule to estimate the probability that two interfaces are aliases (assuming independence between characteristics)

$$\hat{p}_{i,j} = P(i,j \text{ are aliases } | C_1, C_2, ..., C_N)$$
$$= P(C_1 | i,j \text{ are aliases }) ... P(C_N | i,j \text{ are aliases }) P(i,j \text{ are aliases })$$

- Use a set of training data with labeled aliases to learn the likelihood distributions and weights for each characteristic, and the prior probability that a random pair of interfaces are aliases.
- Use the learned distribution to build an alias tree
- Classify interface aliases by thresholds on the interior alias tree node

Alias Tree

- Arranging the interfaces into a tree structure (with interfaces on the leaf nodes), such that the placement on the tree is dependent on the confidence that the leaf nodes are aliases

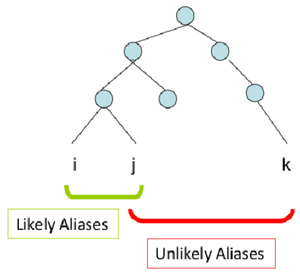

FIGURE 6: Alias tree

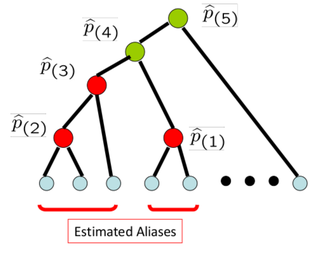

FIGURE 7: Tree-based Alias Classification

- **Construction**
  1. Estimated pairwise alias probabilities for the N interfaces
  2. Get the largest estimated pairwise alias probability and merge them (in-degree, out-degree, hop counts) as a new internal node
  3. Recalculate the alias probabilities associated with this new node
  Repeat (2,3) until all nodes are merged

Experiments

- **Traceroute datasets:**
  1. **Akamai** - 784000 traces (21399 unique IPs)
  2. **Caida**[YHL] - 1800671 traces (552724 unique IPs)
- **Ground-true:**
  1. **Iplane** - 830567 alias pairs (237946 unique IPs)
  2. **Merlin** [Mé10] - 11860197 alias pairs (400040 unique IPs)
- **5-way cross validation** (train on 80%, test on 20%)

Results

- **Inferred distributions to Iplane x Akamai**

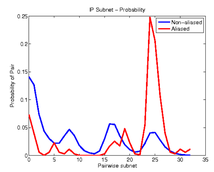

FIGURE 8: Subnet

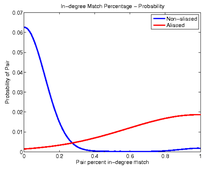

FIGURE 9: In-degree

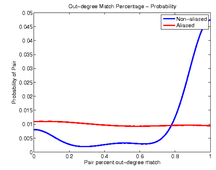

FIGURE 10: Out-degree

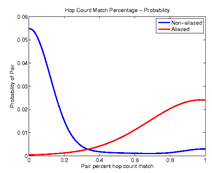

FIGURE 11: Hop Match

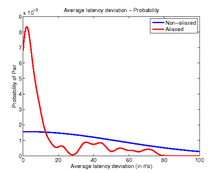

FIGURE 12: Latency Deviation

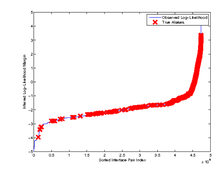

FIGURE 13: Inferred Log-Likelihood

- **Aggregated Detection Rate**

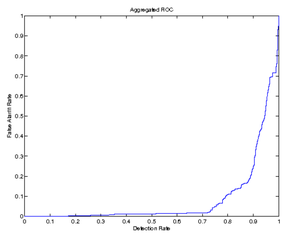

FIGURE 14: Detection Rate: Caida x Merlin (N=30000)

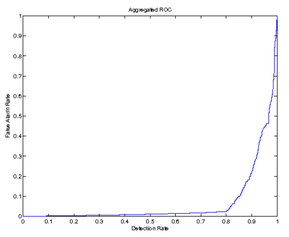

FIGURE 15: Detection Rate: Caida x Iplane (N=30000)

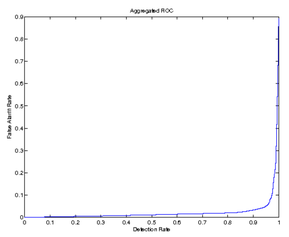

FIGURE 16: Detection Rate: Akamai x Merlin (N=21000)

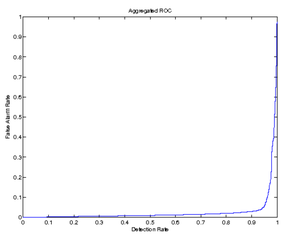

FIGURE 17: Detection Rate: Akamai x Iplane (N=21000)

References

[Mé10] Pascal Mérindol. Merlin <dates used>, 2010.

[YHL] Dan Andersen Emile Aben Young Hyun, Bradley Huffaker and Matthew Luckie. The caida ipv4 routed /24 topology dataset - <dates used>.