Interface disambiguation: a non-probe statistical methodology

Larissa Spinelli, Mark Crovella, Brian Erickson
E-mails: {lspinelli, crovella, erickson}@cs.bu.edu

Abstract

Traceroute probing is a well-known method for discovering links between interfaces on Internet routers. Since 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 methodology. Our results indicate that it is possible to perform highly accurate IP alias resolution without probing.

Motivation

• **Traceroute:** $a \rightarrow w, a, b, r, t, u$
  
  $a \rightarrow p, a, c, k, m, o, p$
  
  $a \rightarrow l, a, c, e, f, h, j, l$

  **Figure 1: Router Graph**

  **Figure 2: Traceroute**

  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)

  $P_{\hat{i},j} = P_{0}(i,j) \text{ are aliases } / (C_1, C_2, \ldots C_N)$

  $= P(C_1(i,j) \text{ are aliases }) \times \cdots \times P(C_N(i,j) \text{ are aliases }) P_{0}(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

  **Interface Disambiguation Methodology**

  **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 - 78,400 traces (21,399 unique IPs)
    
    2. Caida(YHL) - 180,671 traces (52,2724 unique IPs)

  • **Ground-true:**

    1. Iplane - 83,056 alias pairs (23,7946 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**

  **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**

  
  [YHL] Dan Andersen Emile Aben Young Hyun, Bradley Huffaker and Matthew Luckie. The caida ip24 routed /24 topology dataset <dates used>.