Evaluating the Composition and Performance of the IPv6 Address Space with Traceroutes and Web Requests

Our Goal:

Our project focused on examining and evaluating the IPv6 address space, and its various attributes, such as page load times and throughput. The objective of our work was to gain a better understanding of how popular websites and autonomous systems are adopting IPv6, and whether the implementation of this new protocol will affect the current Internet experience in any meaningful way for users and content providers. Our method involved crawling a large dataset of top websites and using a series of object requests to collect statistics on adoption of IPv6, and the resultant performance difference. The results of our work showed that only 15% of the top sites in the dataset implement the new IPv6 protocol, and among those that do, the performance is mostly similar to, or sometimes slightly slower than IPv4.

Methodology:

Using a dataset containing the list of top Alexa sites across fifteen countries, we employed the following methodology to obtain statistics relevant to our inquiry.

1. We began by accessing the sites in the dataset via a Python script.
2. The script crawled through the page to find objects, like images.
3. Another script cleaned these objects and stored them.
4. These objects were then requested using IPv4 and IPv6
5. Step 4 was repeated many times for each site, and the the resultant connection information stored

A breakdown of the code base implementing the methodology:

**parser.py:** this script takes in a list of websites and outputs a JSON that contains a list of aggregated website names from every country.

**associator.py**: this script takes the previous script’s JSON as input, and visits every domain name listed. For each domain, it scrapes the site for objects (images, etc) that we can use for future requests. A JSON of domains and associated objects is outputted.

**cleaner.py:** this script takes in the JSON output by associator.py, and “cleans” up the objects that were found and associated with every domain. To make future requests easy, the script goes through each domain, and finds and builds an easily requestable object. The script labels the best (least likely to cause errors) object, and labels it as the “preferred” key. This association of domain and preferred object is then output into another JSON.

**analyzer.py:** this script runs the experiment. It takes in the JSON generated by cleaner.py, and executes a manual DNS lookup, and then times the requests. The DNS lookup provides information about IPv6 support, and then the timing experiment will use those statistics to decide whether or not to even attempt an IPv6 socket connection. A list of the request timings is then output as the final JSON.

**runner.py:** this script is a wrapper that allows us to run all the scripts sequentially with appropriately labelled output.

**grapher.py:** this script takes in a JSON of domain names with associated information about the attributes of the IPv4 and IPv6 requests, like the time and size of each. Using this information, graphs and visualizations are generated.

Results

Our results showed low adoption of IPv6 among the top sites in our dataset. Only 64 of 422 (15.1%) of sites examined were found to implement the new protocol.