CDN Latency

CDN Latency

Chris, Nilou and Andres

University of California, Riverside

May 27, 2016

Agenda

1. Introduction
2. Methodology
   - CDNs collection
   - Choosing candidates
   - Collecting ping data
   - Building the graph
   - Visualization
3. Results

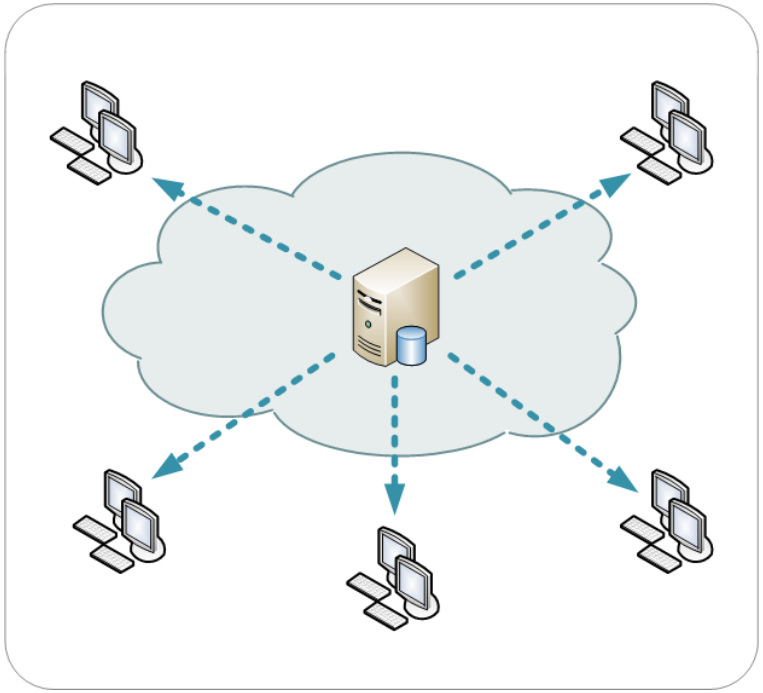
Agenda

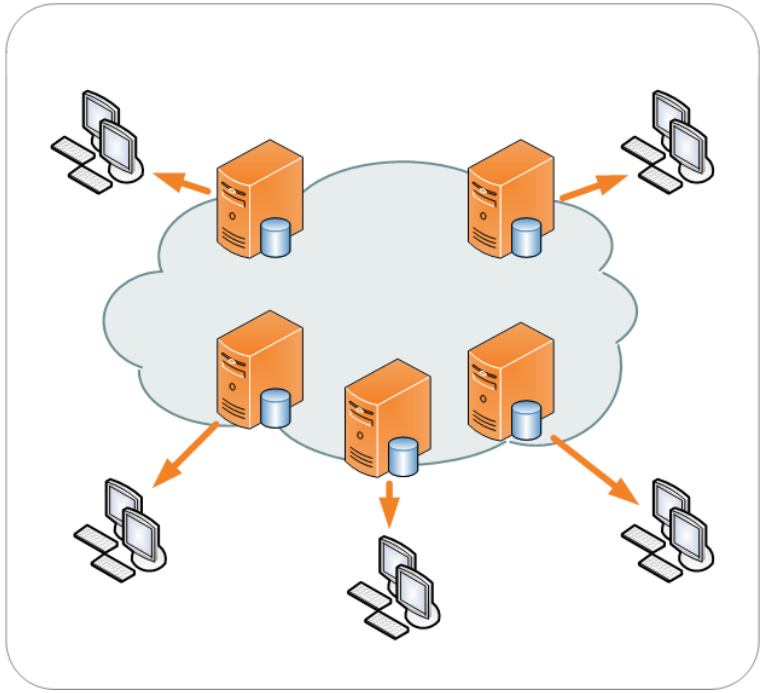
1. Introduction
2. Methodology
   - CDNs collection
   - Choosing candidates
   - Collecting ping data
   - Building the graph
   - Visualization
3. Results

Introduction

- Measuring and Evaluating Large-Scale CDNs (Huang et al., 2008).
- The goal of a CDN is to serve content to end-users with high availability and high performance.

Introduction





Introduction

- But... Where do we deploy a new CDN server???

Agenda

1 Introduction

2 Methodology
- CDNs collection
- Choosing candidates
- Collecting ping data
- Building the graph
- Visualization

3 Results

Overview

1. Collect information about current CDNs
2. Select a group of possible candidates
3. Measure the latency between points (Ping time)
4. Graph analysis
5. Visualization

Agenda

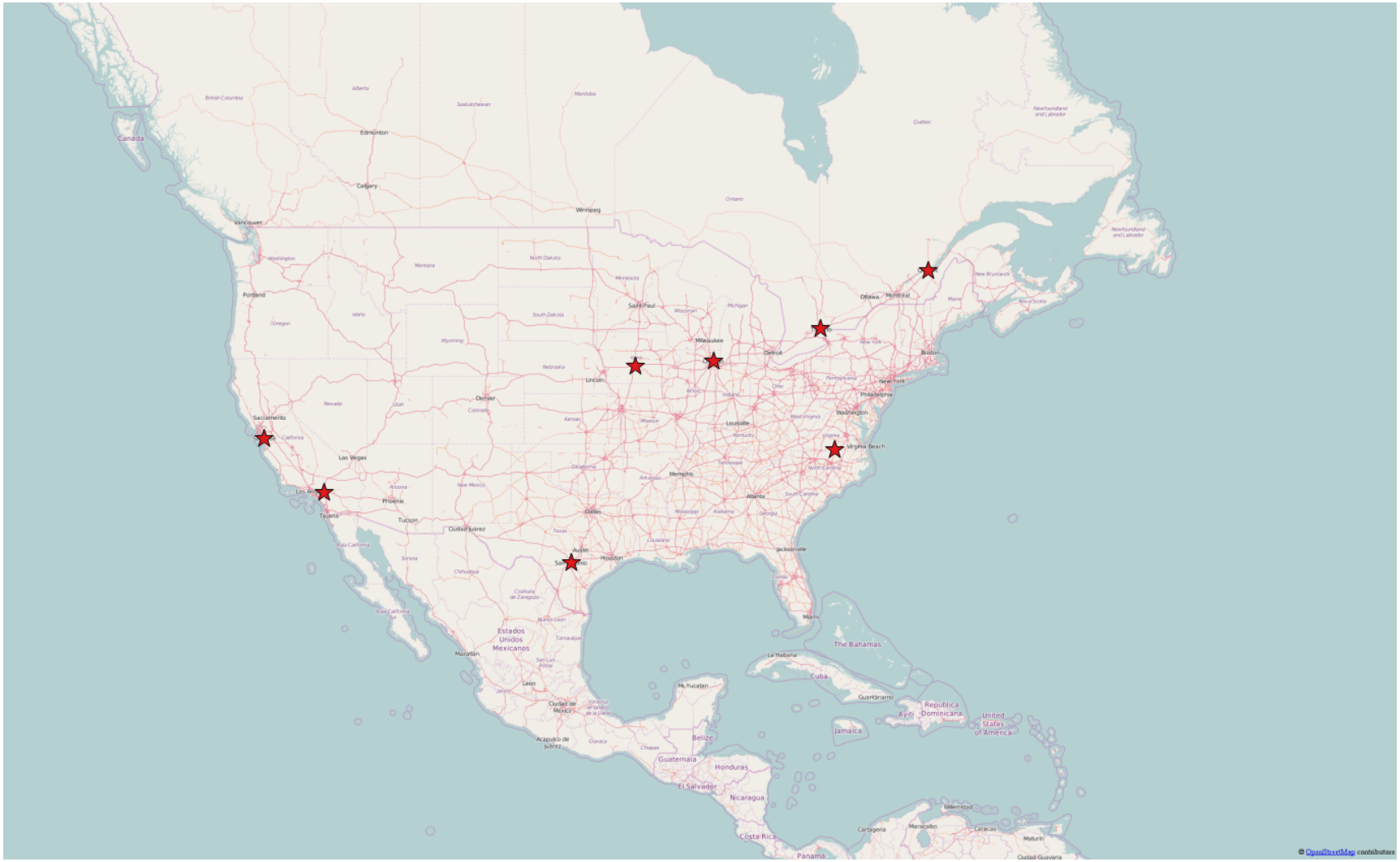
1. Introduction
2. Methodology
   - CDNs collection
   - Choosing candidates
   - Collecting ping data
   - Building the graph
   - Visualization
3. Results

Virtual machines

- Microsoft Azure: Cloud Computing Platform & Services [1]
- 8 instances in different geographical regions
  1. East USA
  2. Central USA
  3. South Central USA
  4. West USA
  5. North Central USA
  6. East Canada
  7. Central Canada
  8. Riverside

[1] https://azure.microsoft.com/

Virtual machines



Agenda

1. Introduction
2. Methodology
   - CDNs collection
   - Choosing candidates
   - Collecting ping data
   - Building the graph
   - Visualization
3. Results

Choosing candidates

- Universities Worldwide [2]
- Universities from USA and Canada

[2] http://univ.cc/

Choosing candidates

- Cleaning and processing
  - Remove urls which did not respond to ping
  - Remove duplicates (in the same city)
  - Remove incorrect format
  - Extract latitude and longitude (reverse geocoding)
- End up with 97 candidates

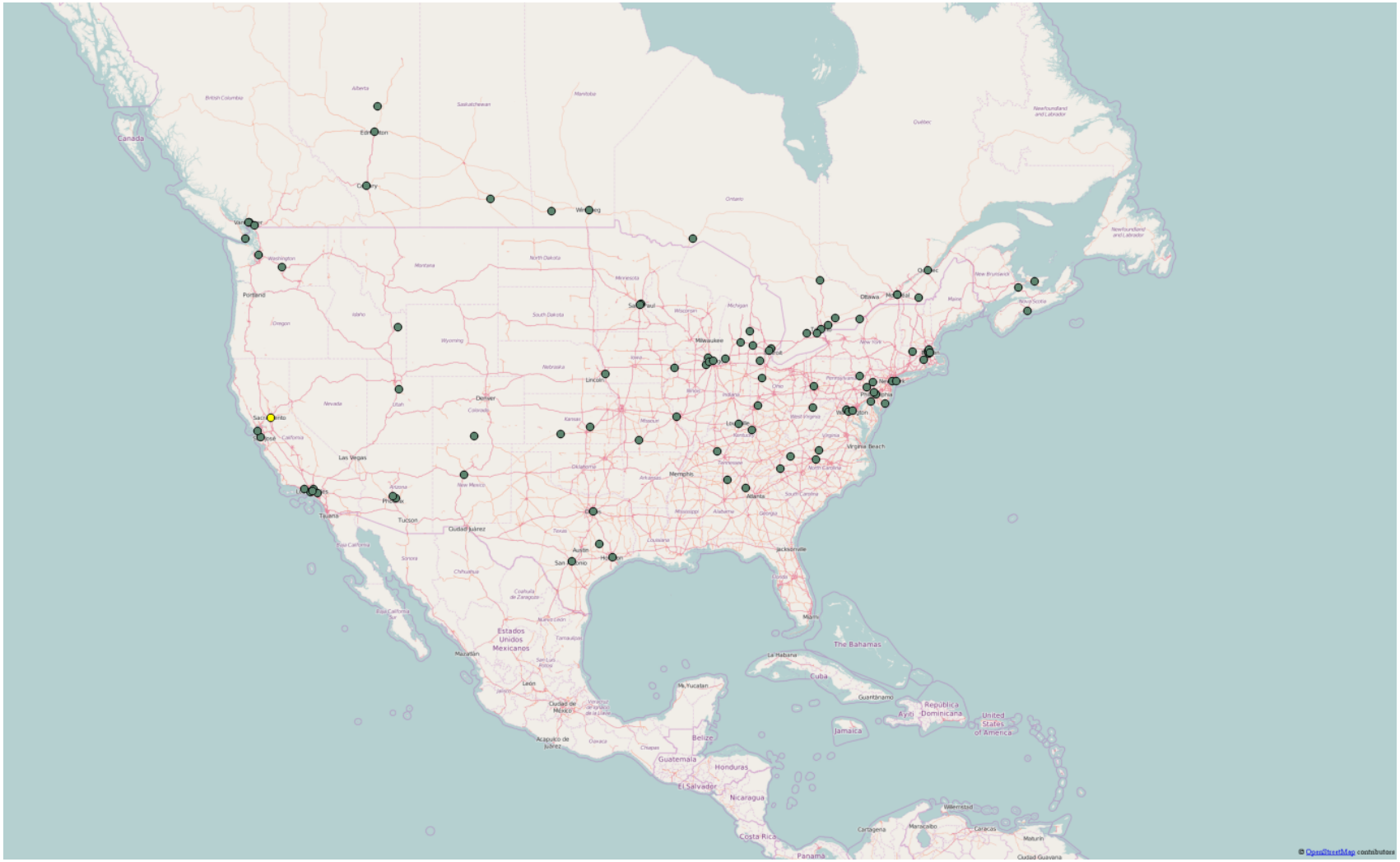
Python scripts

```
import xml.dom.minidom
import socket
from geoip import geolite2
import subprocess

doc = xml.dom.minidom.parse("usa.xml")
list = doc.getElementsByTagName("a")
for item in list:
        url = item.getAttribute("href")[7:-1]
        try:
                addr = socket.gethostbyname(url)
                match = geolite2.lookup(addr)
        except socket.error, msg:
                addr
        else:
                try:
                        output = subprocess.check_output(["ping","-c1",url])
                except subprocess.CalledProcessError,e:
                        url
                else:
                        print "{0}".format(url)
```

Python scripts

```
import socket
from geoip import geolite2
import subprocess
import geocoder
import time

f = open('dns3.txt', 'r')
for url in f.readlines():
        try:
                url = url[:-1]
                addr = socket.gethostbyname(url)
                match = geolite2.lookup(addr)
                lat = round(match.location[0],2)
                lon = round(match.location[1],2)
                g = geocoder.google([lat,lon], method='reverse')
                time.sleep(2)
        except socket.error, msg:
                print ""
        else:
                print(url+";"+addr+";"+str(lat)+";"+str(lon)+";"+g.city+";"+g.state)
```

Choosing candidates



Agenda

1. Introduction
2. Methodology
   - CDNs collection
   - Choosing candidates
   - Collecting ping data
   - Building the graph
   - Visualization
3. Results

Ping data

- We collect ping data from each current CDN to all the candidates
- Run a script at each virtual machine
- Transmit 5 packages and extract the average ping time

Python script

```
import subprocess

f = open('dns3.txt', 'r')
for url in f.readlines():
        try:
                url = url[:-1]
                output = subprocess.check_output(["ping","-c5",url])
        except subprocess.CalledProcessError,e:
                url
        else:
                ping = float(output.split('\n')[-2:-1][0].split('/')[-3:-2][0])
                print "{0};{1}".format(url, ping)
```

Agenda

1. Introduction
2. Methodology
   - CDNs collection
   - Choosing candidates
   - Collecting ping data
   - Building the graph
   - Visualization
3. Results

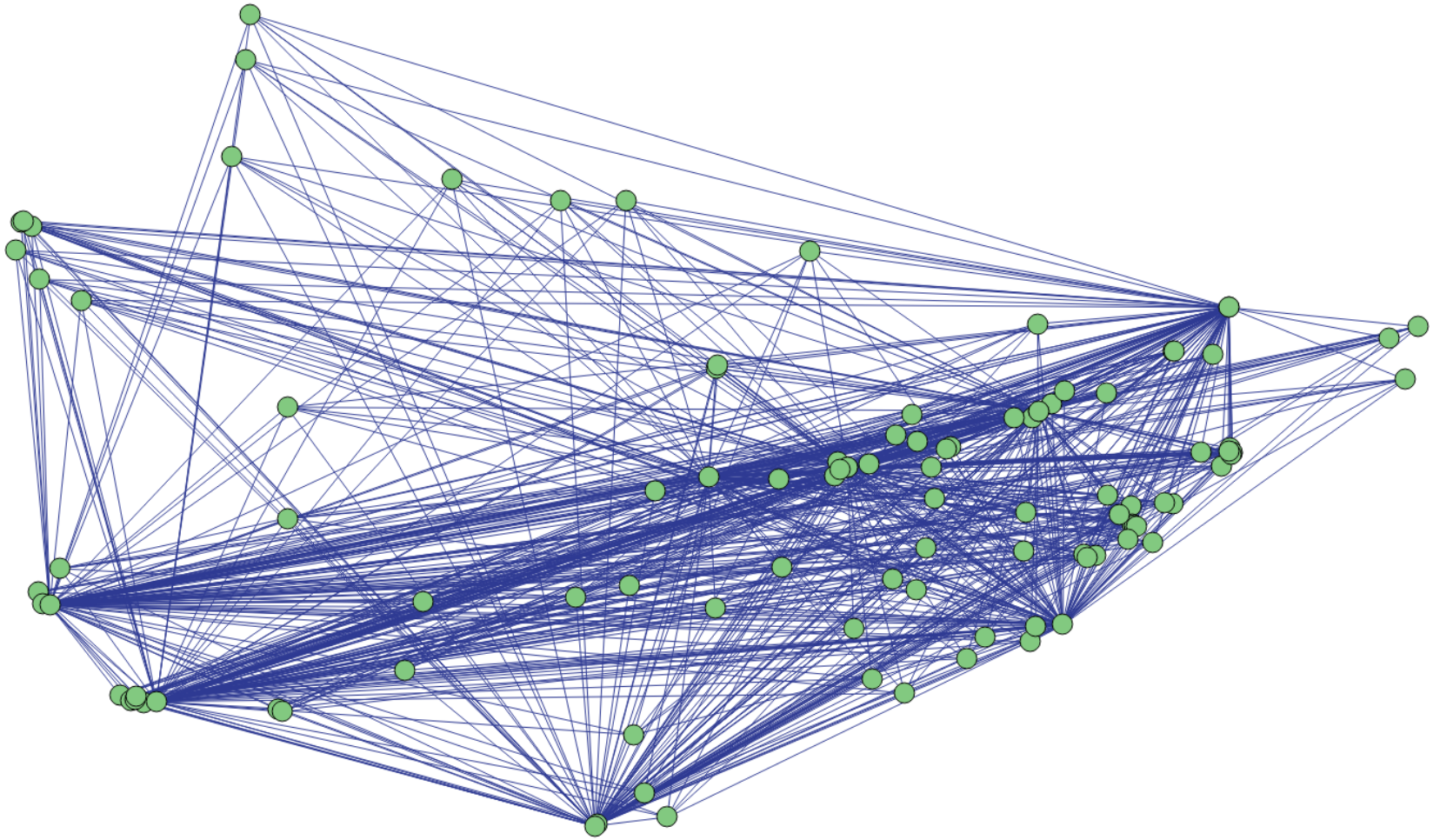
Building the graph

- Merge results for each virtual machine (CDN)
- Use `networkX` package in python
- Final graph has 105 nodes and 776 edges

Python script

```
import networkx as nx
import pandas as pd

G=nx.Graph()
## Riverside
center_y = 33.9680
center_x = -117.3194
center = (center_x,center_y)
G.add_node(center)

candidates = pd.read_csv('candidates.csv', sep=';', header=None
                                    , names=['url','ip','lat','lon','city','state'])
pings = pd.read_csv('pings.csv', sep=';', header=None, names=['url','ping'])

nodes = candidates.loc[:,['url','lat','lon']].merge(pings).loc[:,['lat','lon','ping']]

for index, node in nodes.iterrows():
        point = (node['lon'],node['lat'])
        G.add_node(point)
        G.add_edge(center,point,weight=node['ping'],cdn='Riverside')
```

Building the graph



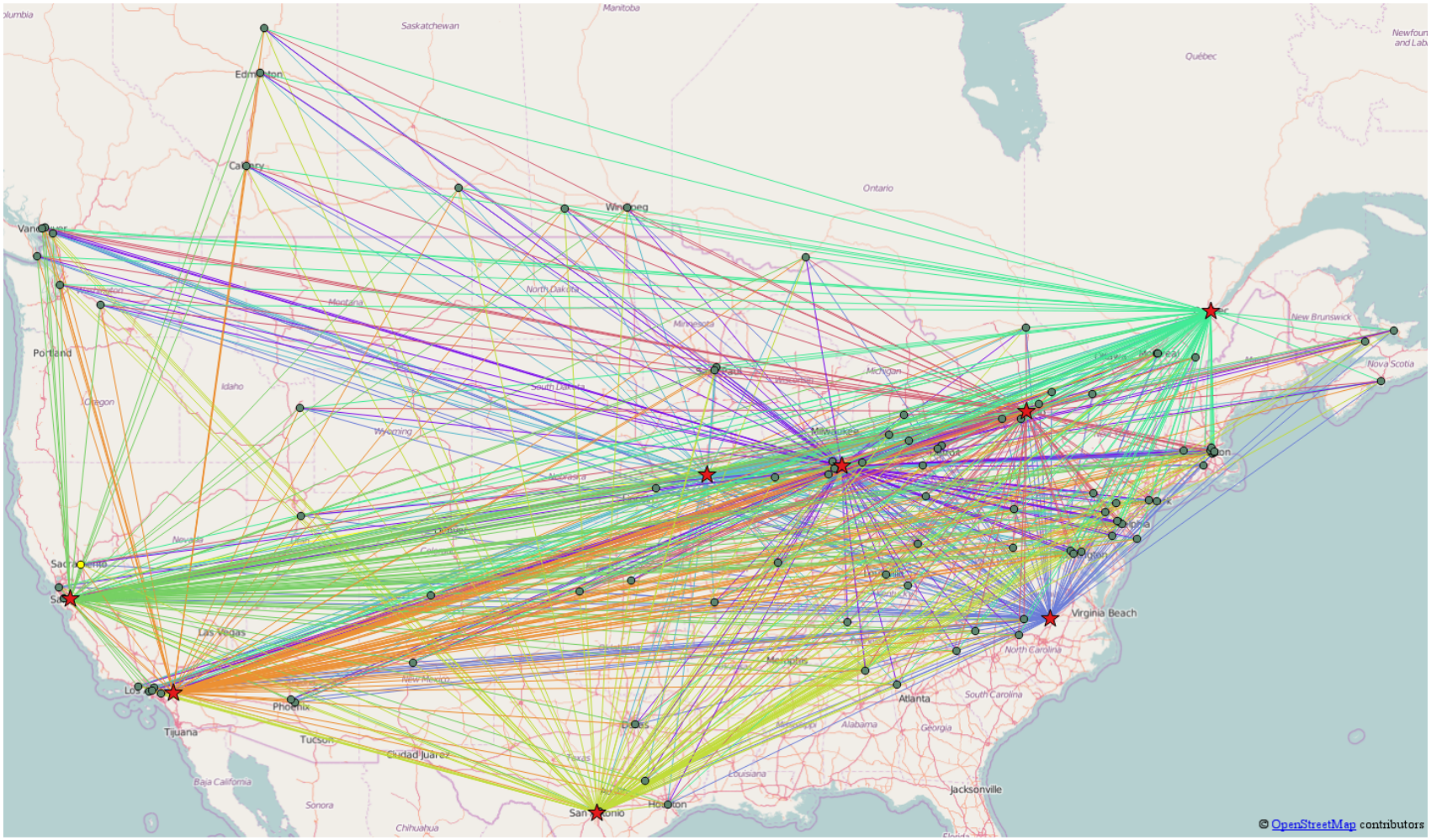
Agenda

1. Introduction
2. Methodology
   - CDNs collection
   - Choosing candidates
   - Collecting ping data
   - Building the graph
   - Visualization
3. Results

Visualization

- Use networkX to save the output to a shapefile format
- Use QGIS to visualize data

Visualization



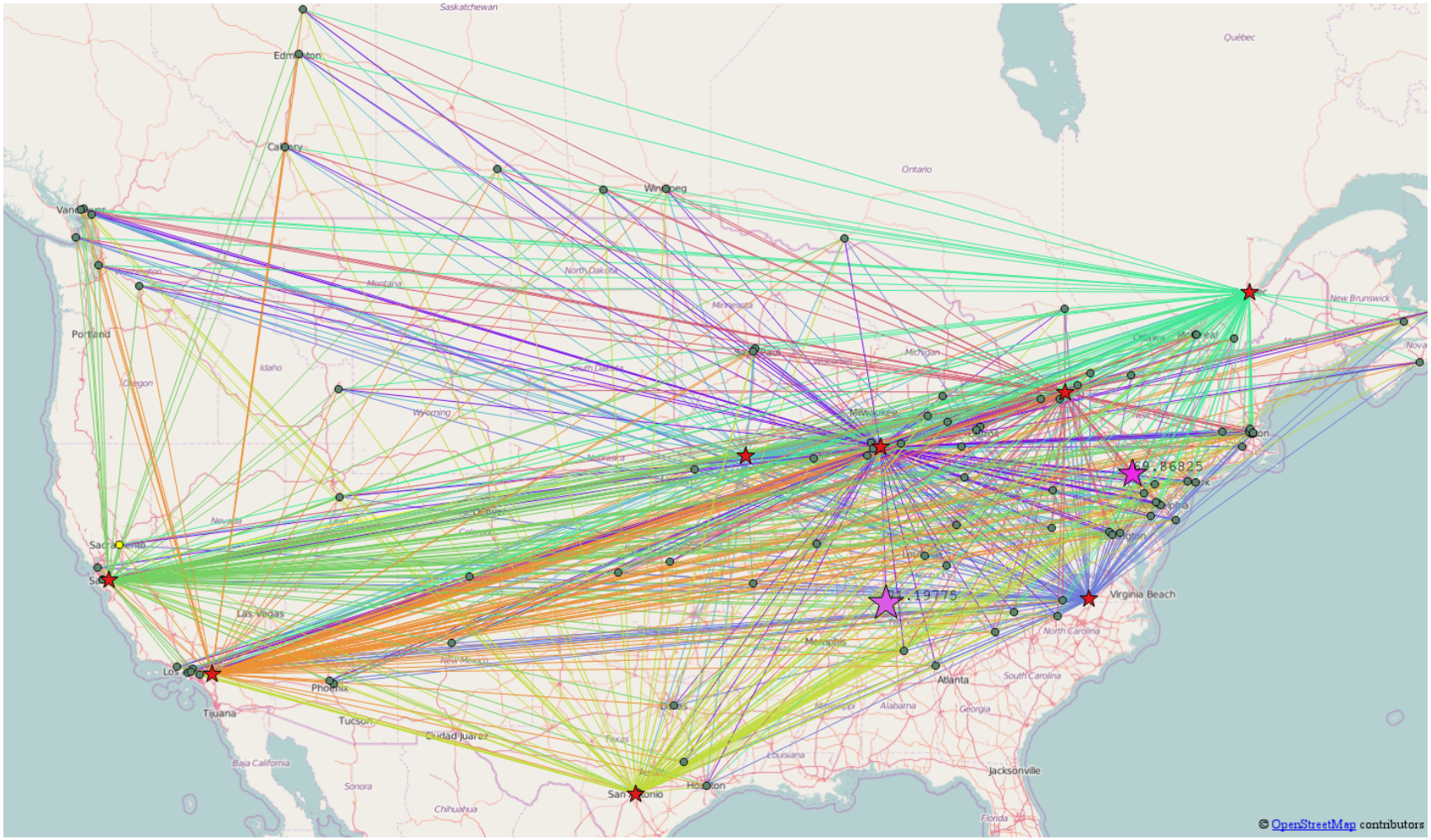
Agenda

1. Introduction
2. Methodology
   - CDNs collection
   - Choosing candidates
   - Collecting ping data
   - Building the graph
   - Visualization
3. Results

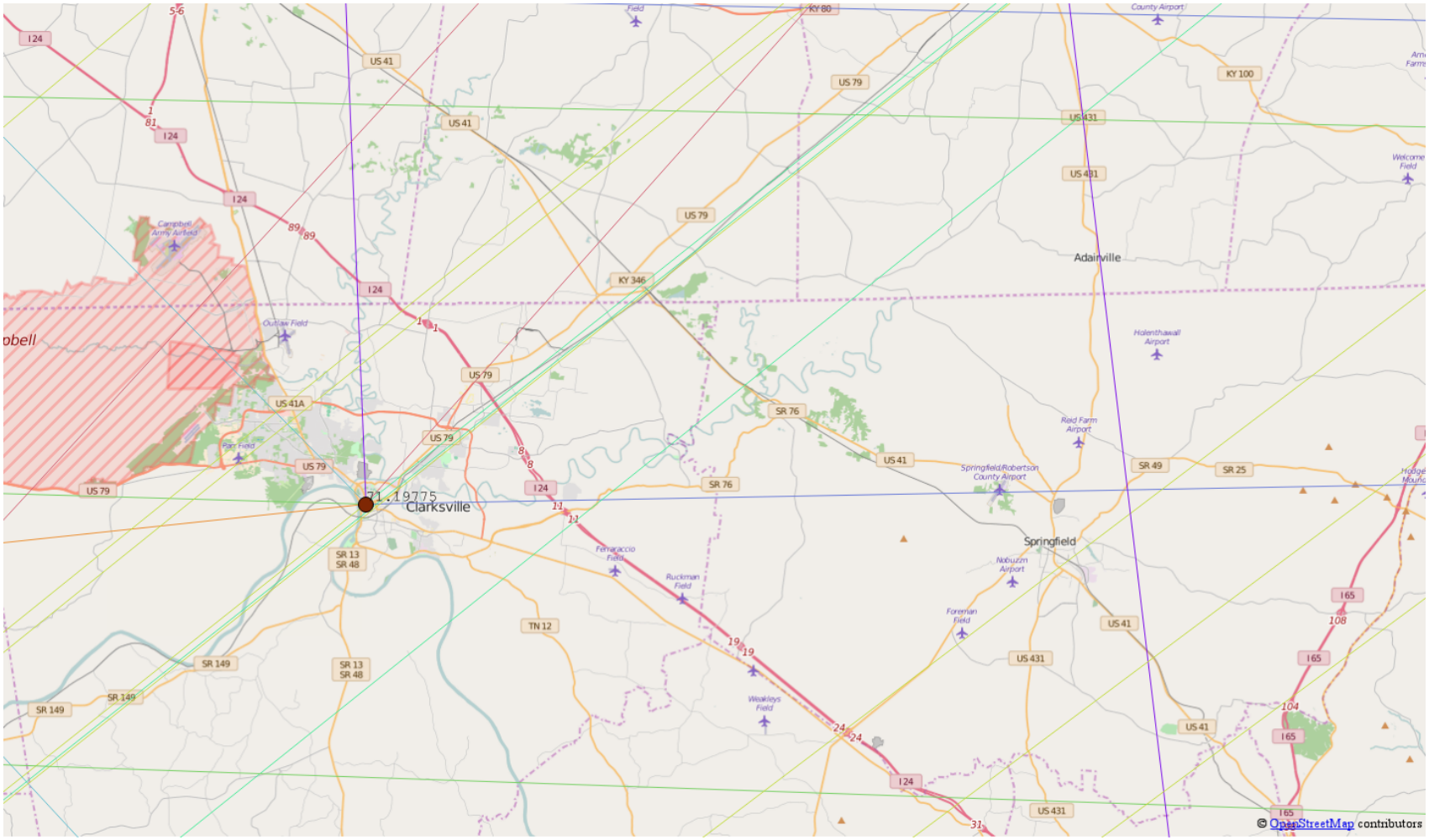
Results

- Iterate at each candidates to all the CDNs
- Compute the average ping time
- Sort by average time ascendantly

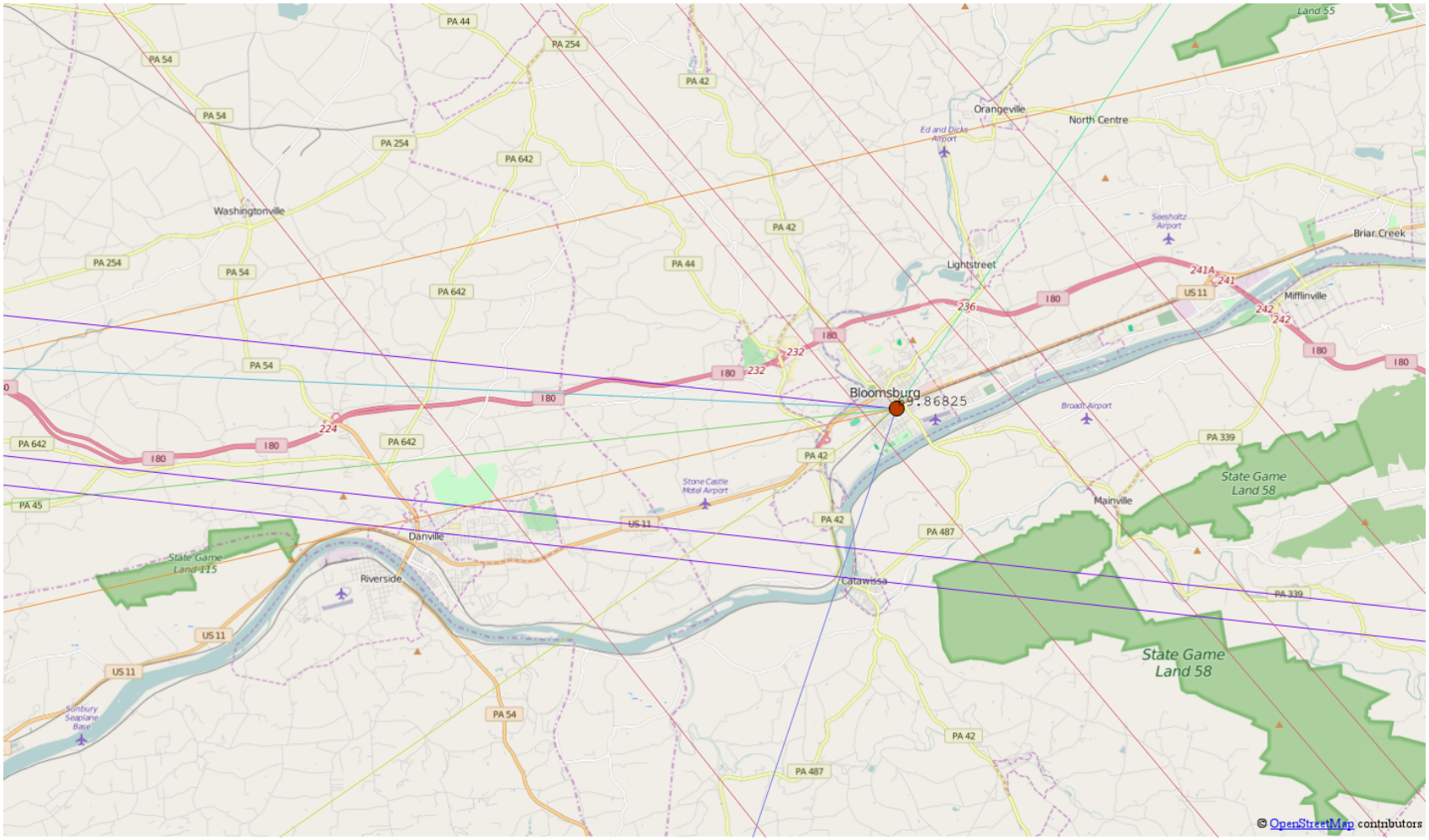
Results


Saskatchewan
Québec
Ontario
Edmonton
Calgary
Winnipeg
Vancouver
Portland
Oregon
Idaho
Montana
North Dakota
Minnesota
Wyoming
Michigan
New Brunswick
Nevada
Las Vegas
Phoenix
Tijuana
Tucson
Ciudad Juárez
Baja California
Sonora
Chihuahua
Texas
Louisiana
Alabama
Georgia
Mississippi
Atlanta
South Carolina
North Carolina
Virginia Beach
Jacksonville
Florida
69.86825
71.19775
© OpenStreetMap contributors

Results


Clarksville
Springfield
Adairville
US 41
US 79
I 24
KY 80
KY 100
US 431
KY 346
SR 76
SR 49
SR 25
SR 13
SR 48
SR 149
TN 12
I 65
US 41A
© OpenStreetMap contributors

Results


Bloomsburg
Orangeville
North Centre
Lightstreet
Briar Creek
Mifflinville
Mainville
Danville
Riverside
Catawissa
Washingtonville
State Game Land 58
State Game Land 115
PA 54
PA 254
PA 42
PA 44
PA 642
PA 45
PA 339
PA 487
I 80
US 11
224
232
236
241
241A
242
© OpenStreetMap contributors

Thank you!!!