The Current State of the Art in Network Measurement

David Clark
MIT

Matthew Luckie
CAIDA

kc claffy
CAIDA

August 20, 2025

Abstract

Measurement of the Internet is a critical part of understanding how it is operating. Measurement can inform network operators, individual users, researchers across a range of disciplines from political science to security, and the policy community. For example, individuals perform speed tests to see if they are getting the broadband service they think they should be getting. Measurements made by the technical measurement community are more complex: they measure latency, topology and routing, security abuses, and service impairments and outages, among other things.

With the evolution of modern research infrastructure and open-source software tools and data, scientific measurement of the Internet is no longer limited to experts. The latest generation of tools on CAIDA's Ark research infrastructure support powerful tests – such as measuring latency from hundreds of global points to a target server – with as few as 15 lines of Python.

Finding existing data can be another hurdle. Projects like the Internet Yellow Pages address this by aggregating over 50 measurement datasets for integrated public access.

In this paper, we share example results from the Ark and IYP platforms, show how to use them, and encourage collaboration between TPRC participants and the Internet measurement community.

1 Introduction

The Internet measurement community is active, with results presented at high-quality technical conferences. Methods range from purpose-built infrastructure to opportunistic use of other data. While some findings are relevant to policy-makers, economists, and advocates, direct engagement with these communities is rare. Barriers to engagement include limited technical fluency among policy audiences and a lack of visibility into what measurement experts can do. As a result, policy researchers often must discover and interpret technical work on their own. Publication in technical conferences that do not focus on applica-

tion to policy questions further limits accessibility. Cross-disciplinary exchange barriers are often insurmountable (Section 8).

This paper aims to lower that barrier by describing the state of the art in network measurement and data curation. Recent advances make it possible to run sophisticated global tests in minutes—without deep technical expertise—and to locate existing datasets more easily.

We address two key challenges for non-technical researchers:

1. How easy is it for them to run their own measurement campaigns?
2. How can they find and query relevant existing data?

Our goal is to spark cross-disciplinary collaboration, enabling more researchers—and perhaps their graduate students—to run or co-run Internet measurements themselves. By reducing the time needed to launch global experiments from *months to minutes*, we aim to make Internet measurement research accessible well beyond the circle of technical experts.

2 Types of measurements

Internet measurement methods fall broadly into two categories: *passive* and *active*.

Passive measurement involves observing and recording traffic that flows across an Internet link, then analyzing it later. This type of measurement can provide rich detail about actual user behavior, traffic patterns, and protocol usage. However, it faces significant technical hurdles, such as the need for high-capacity capture systems, and serious privacy concerns, since it involves real user data. Section 7 describes some passive measurement activities but this paper's main focus is on active measurement.

Active measurement works by sending specially crafted packets into the Internet and analyzing how the network responds. This approach can reveal key properties such as network topology, latency, and throughput (as in speed tests). Active methods can also target specific Internet services,such as components of the Domain Name System (DNS), to assess their availability, performance, or configuration.

A third category of measurements does not fit neatly into either passive or active. Most notably, researchers often collect and analyze the state of control protocols like the Border Gateway Protocol (BGP), which governs global Internet routing. Although routing data can be highly revealing about network connectivity and events, we will not examine it in detail here.

3 Implementing active measurement

Active measurement requires devices—called vantage points (VPs) or probes—placed at the network edge to initiate tests. One approach is *crowd-sourced measurement*, where users run tests from their own computers. The most common

example is speed testing, as with Ookla[1], MLab NDT[2], and Cloudflare [3]. People run these tests to check if they're getting the performance they paid for or to troubleshoot slow connections. However, crowd-sourced data has several limitations for research. First, researchers cannot genearlly control when measurements occur, as they depend on user initiation. Second, users often run tests when problems occur, biasing results toward anomalies rather than normal conditions. Third, researchers cannot design a specific measurement campaign; they must work with whatever data has been collected. Because of these constraints, this paper does not focus further on crowd-sourced data.

An alternative to crowd-sourced data is to deploy *dedicated VPs* programmed specifically for active measurement. Examples include CAIDA's Ark (Archipelago) platform (our focus in this paper), and RIPE Atlas, which supports a more limited set of measurements but has broader deployment. While Ark has fewer probes than RIPE Atlas, it supports a wider variety of basic probing functions that can be tailored and combined into complex experiments.

3.1 Balancing Capability and Safety

Active Internet measurement is not risk-free. Poorly designed experiments can consume excessive resources, violate laws, or cause reputational harm. For example, probing content blocked in a jurisdiction by downloading prohibited material could create legal or diplomatic issues.

Ark's design balances capability with safety. Instead of giving users raw packet-sending ability, Ark offers a Python-based programming environment with predefined measurement primitives. These primitives:

1. Clearly define the type of traffic each VP will send.
2. Allow Ark operators to constrain permissible experiments.
3. Help VP hosts understand exactly how their vantage points will be used and what risks are involved.

For more on the Ark programming environment's design choices, see Luckie *et al.*'s, An Integrated Active Measurement Programming Environment [1]. Portions of the explanatory material we present are derived from that paper.

Design goals that shaped the Ark interface are:

- **Ease of use:** Measurement capabilities are exposed via Python, a widely used language in both academia and industry. Researchers can run tests and receive results as Python objects, with many reusable libraries available.
- **Performance:** Low latency between measurement and results enables responsive, reactive experiments.

[1] https://www.speedtest.net/
[2] https://speed.measurementlab.net
[3] https://speed.cloudflare.com/

- **Interoperable and Extensible:** Ark's software runs on diverse operating systems, works in containers, has minimal optional dependencies, and is distributed in packaged form for easy deployment.

Ark probes can perform:

- `ping`: measure latency to an IP address.
- `traceroute`: discover the sequence of router interfaces to a target IP.
- **DNS lookups**: resolve hostnames to IP addresses.
- **HTTP GET**: fetch web pages.
- **more advanced tests**, e.g., multi-step probing and service-specific measurements.

These capabilities are programmed in Python, allowing experimenters to combine primitives into sequences. For example, a script could perform a DNS lookup to resolve a hostname, then ping the resulting IP.

Programs are short and expressive. For instance, measuring latency from every Ark probe to a target IP and reporting the minimum can be done in a few lines of code, which we reproduce below. One could extend such a script into a geolocation tool using triangulation.

```
import sys
from datetime import timedelta
from scamper import ScamperCtrl

if len(sys.argv) != 2:
  print("usage: shortest-ping.py $ip")
  sys.exit(-1)

mux = "/run/ark/mux"
with ScamperCtrl(mux=mux) as ctrl:
  ctrl.add_vps(ctrl.vps())
  for i in ctrl.instances():
    ctrl.do_ping(sys.argv[1], inst=i)

  min_rtt = None
  min_vp = None
  for o in ctrl.responses(timeout=timedelta(seconds=10)):
    if o.min_rtt is not None and (min_rtt is None or min_rtt > o.min_rtt):
      min_rtt = o.min_rtt
      min_vp = o.inst

  if min_rtt is not None:
```

```
    print(f"{min_vp.name} {(min_rtt.total_seconds()*1000):.1f} ms")
  else:
    print(f"no responses for {sys.argv[1]}  to that address")
```

This is not just a code *snippet*–it's a complete, runnable program. In only four lines of code, we instruct every Ark unit to ping the target IP address, and in another four lines, we retrieve the responses.

For readers familiar with Python:

- Each value returned by `ctrl.instances()` identifies a single probe.
- `mux` is the Unix channel used to launch the executions.
- All probes run in parallel, and each execution returns a `response`, which is processed as soon as it arrives.
- A timeout is set to protect against probes that fail to respond, preventing the execution from hanging.

At first glance, the value of these measurement primitives may not be obvious. However, researchers have used the platform to investigate a wide range of important questions, such as:

- **Global latency to a specific URL** – What is the latency from each Ark node to a specific URL?
- **Path diversity and security** – When reaching a given URL from different parts of the globe, how many Autonomous Systems (ISPs) does the path cross? This relates to resilience and resistance to route hijacking, among other concerns.
- **Web hosting distribution** – For popular websites in various countries, how many are hosted locally, how many use multiple hosting locations to provide low-latency access worldwide, and how many are hosted entirely outside the country?
- **Latency variation over time** – For a given network path, does latency vary over a day or a week? Such patterns can reveal potential congestion.
- **Anycast deployment patterns** – Some network services use anycast, where multiple servers share the same IP address and routing directs users to the nearest instance. What does the global deployment look like? Which probes reach which servers, and what patterns emerge?

4 A more complex example–probing to a URL

In the previous example, we used `ping` to find the shortest round-trip time (RTT) to a specific IP address. In practice, however, we may not know the

exact IP address in advance. Appendix 4 presents a program that takes a URL as input, performs a DNS lookup from every probe, and retrieves the IP address for that location. Because the same hostname often resolves to different IP addresses in different regions of the world, this approach reveals geographic variation in resolution. The data below shows the results when the target URL is www.cnn.com.

syd3-au AU NSW 0.13
bom2-in IN 0.17
lax6-us US CA 0.2
mel3-au AU VIC 0.2
syd4-au AU 0.21
lwc-us US KS 0.29
fra-de DE HE 0.3
dub-ie IE 0.33
sin-sg SG 0.36
scl4-cl CL 0.37
yto-ca CA ON 0.38
fra2-de DE 0.41
mia3-us US FL 0.43
sto-se SE 0.47
lga-us US NY 0.49
sin5-sg SG 0.49
nrt-jp JP 0.5
sjc4-us US CA 0.55
cdg4-fr FR 0.55
arn-se SE 0.59
atl7-us US GA 0.6
ams9-nl NL NH 0.65
nrt4-jp JP 0.66
lhr-uk UK 0.69
dfw2-us US TX 0.78
ord-us US IL 0.84
mad3-es ES 0.87
sjc6-us US CA 0.89
atl4-us US GA 0.89
lax4-us US CA 0.9
jfk2-us US NY 0.9
ory8-fr FR 0.94
lgw2-uk UK 0.95
sao3-br BR SP 1.03
akl-nz NZ 1.17
ewr-us US NJ 1.23
wbu-us US CO 1.35
ord2-us US IL 1.49
del-in IN 1.67
akl2-nz NZ 1.75
aep2-ar AR 1.75
nrn-nl NL 1.91
cgs-us US MD 2.08
hkg5-cn CN 2.1
nrt3-jp JP 2.28
vie-at AT 2.47
hnd-jp JP 2.65
mad2-es ES 2.66
sna-us US CA 2.73
san-us US CA 2.8
cys-us US WY 2.88
san9-us US CA 3.29
hlz2-nz NZ 3.56
mnz-us US VA 3.88
adl-au AU 4.07
ygk-ca CA ON 5.51
san2-us US CA 5.53
bfi-us US WA 5.67
cld6-us US CA 5.92
man-uk UK 6.32
cjj-kr KR 6.37
sdm-us US CA 6.74
blq-it IT 7.51
dus-de DE NRW 7.98
san8-us US CA 8.36
rdu2-us US NC 8.52
syr-us US NY 9.43
ams8-nl NL NH 9.61
las-us US NV 9.7
scq-es ES 10.45
zrh3-ch CH ZH 11.51
smf2-us US CA 11.6
aep3-ar AR 11.64
drs-de DE SN 12.1
okc2-us US OK 12.94
abz2-uk UK IN 13.24
pna-es ES 13.44
cld5-us US CA 13.8
nap2-it IT 14.06
bwi3-us US MD 14.65
ceu-us US SC 15.2
blr2-in IN 15.42
ind2-us US IN 15.53
bio-es ES 15.59
lex-us US KY 15.71
waw-pl PL 15.81
poa2-br BR 16.05
pvu3-us US UT 16.08
svo-ru RU 16.55
pvu-us US UT 17.22
psa-it IT TOS 17.66
waw2-pl PL 18.51
atl5-us US GA 19.1
bed-us US MA 19.33
bre-de DE 19.76
arb4-us US MI 22.66
mex2-mx MX 22.71
rdu3-us US NC 23.96
rap-us US SD 24.74
sjj-ba BA 26.43
mob-us US AL 28.49
bkl-us US OH 29.25
lnk-us US NE 29.74
mia-us US FL 30.28
cou-us US MO 30.32
sea4-us US WA 31.73
icn-kr KR 33.15
beg-rs RS 33.79
hou-us US TX 36.2
boi-us US ID 38.29
tlv3-il IL 40.84
hkg4-cn CN 41.74
tlv4-il IL 46.74
xna-us US AR 47.99
hnl4-us US HI 48.21
oun-us US OK 49.12
acv-us US CA 50.08

dar-tz TZ 55.59	sbd2-us US CA 69.68	kgl-rw RW 152.31
arb3-us US MI 56.11	sjc3-us US CA 70.85	jnb3-za ZA 179.59
lax5-us US CA 63.93	prg3-cz CZ 100.59	sin4-sg SG 239.87
bur-us US CA 65.6	itm-jp JP OS 104.3	
ful-us US CA 69.03	gum-gu GU 123.82	

A close look at these latency measurements reveals unusual patterns. One risk in using this tool is that a researcher unfamiliar with the limits of the experiment might misinterpret the results.

For example, an Ark probe located in a home and connected via a cable (HFC) system will typically have a minimum latency of around 10 ms due to the scheduling algorithm used by HFC networks. In other cases, we see latencies of only a few milliseconds or less. This could indicate one of two things:

- The probe is on a Fiber-to-the-Home (FTTH) connection, which avoids significant scheduling delays.
- The probe is located in a data center, with the target server hosted in the same facility.

The sub-millisecond latency for the first few probes in the list suggests that the probe and the target server are essentially co-located-—a strong clue to the actual location of that copy of the data.

When latencies rise substantially above 10–12 ms, it suggests the nearest copy of the data is farther away. As another reference point, the typical round-trip latency across the continental U.S. is about 70 ms. The very high latencies in the last few entries in the list suggest that further investigation might be warranted. However, these figures should not be interpreted with high precision—there are too many potential confounding factors to treat them as exact.

5 A more complex example–composing different measurements.

This example-—though somewhat involved-—shows how different Ark tests, combined with knowledge of how an underlying system works, can produce valuable insights. Content providers typically deliver data through a Content Delivery Network (CDN), hosting copies of their content in multiple locations. The key question is: *How does the provider choose which location to use for a given client?* Figure 1 uses Ark to measure the Netflix infrastructure to show that Netflix is using latency data to pick the source in their CDN.

Netflix offers a public speed test that directs clients (typically a web browser) to a nearby content server, from which large files are downloaded to measure throughput. For this experiment, no actual speed test was necessary—only the identity and location of the selected server. By tracking how that choice changed with latency, we could infer Netflix's selection logic.

The experiment combined four Ark operations:

- *DNS lookups* to obtain the IP address of the speed test server chosen by Netflix for a given VP.
- *HTTP fetch* to query the Netflix speed test REST API for a list of recommended test servers for that VP.
- *DNS lookups* to resolve the IP addresses of those recommended servers.
- *Traceroute and ping* to determine topological paths and measure performance characteristics.

Ark's integrated platform supports all these operations, enabling the experiment to be implemented and executed entirely within its environment.

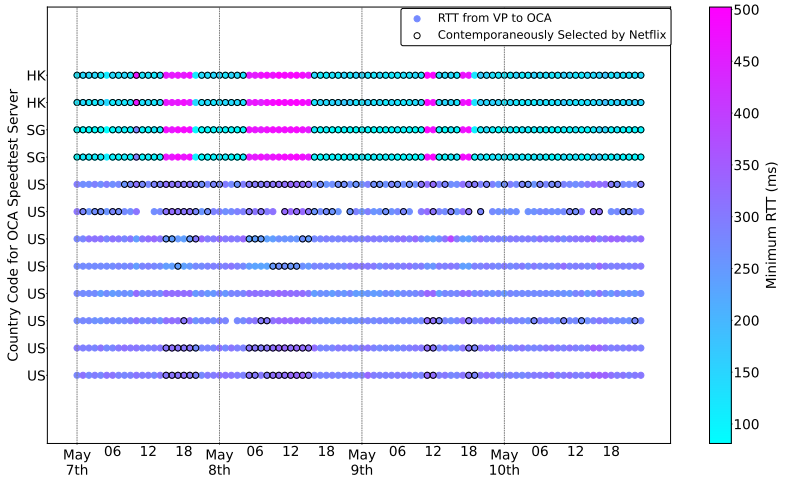

Figure 1: **Effect of latency variation on Netflix server selection, May 2024, for a VP in Thimphu, Bhutan.** The X-axis shows the VP's local time. Each row represents a unique address block, annotated with the server's country. Circles mark which server Netflix selected at each time point. Latencies to Hong Kong and Singapore servers fluctuate widely, and during high-latency periods Netflix switches to a U.S. server. While latency to the U.S. is high, it remains lower than the degraded HK and SG latencies during those times.

We use this example to illustrate the power of the Ark platform for non-technical users. The challenge we want to offer the reader is not to focus on the code, but instead to ask yourself: what sort of interesting queries could be realized by composing these primitives.

6 Finding and querying existing data – Internet Yellow Pages

The second challenge for non-technical researchers is locating existing data. The data needed to address a specific question may already be available, but how can someone outside the field know where to find it? Fortunately, advancements in this area have made data discovery more accessible. We describe a new Internet-based service, the Internet Yellow Pages, developed by researchers at Internet Initiative Japan Inc. (IIJ). This service consolidates over 50 datasets into a single, user-friendly platform for easy exploration.

The Internet Yellow Pages (IYP) offers two primary methods for accessing its data. The first is through the Internet Health Report website.[4] Users can explore available data types and run queries to view specific datasets. Queries can reveal insights into patterns of interconnection among Autonomous Systems (ASs), network latency, congestion, and outages. In the search field of the Internet Health Report, one can enter the number of an AS, an Internet exchange, a network name, an IP address (technically a *prefix* or block of addresses), or a country, to learn what data is available about it.

For more advanced analysis, users can access raw data from over 50 sources. In a traditional database, this data might be organized as records with multiple fields, queried using a language like SQL. However, IYP stores data as a *knowledge graph* in a Neo4j graph database. *Nodes* (e.g., BGPPrefix, Country, DomainName, Facility, HostName,Organization,Prefix, Ranking or URL, among others) are linked by relationships (e.g., DEPENDS ON, LOCATED IN, MEMBER OF, PARENT, PEERS WITH, or RESOLVES TO). Queries for this graph structure are written in Cypher, a language similar to SQL but designed for graphs.

We are not going to give an introduction to Cypher, which requires dedicated study (or a trained LLM) to generate queries in this language. Instead, at the end of the paper we include links to useful resources that can help anyone that wants to play.

To illustrate, we present an example query output. Consider the following query: for a given country, show all the Autonomous Systems (ASes or ISPs) in that country, and then show who the operator is for each AS. Figure 2 shows this information for Qatar as an example.

Such graphs can be useful, but also are a bit of eye candy. If you want to work with the data, Cypher allows you to export the data in several machine-readable formats (CSV and JSON). We show a tabular form in Table 1.

While we are not providing a tutorial on how to use the IYP knowledge graph, to produce the data in that table takes one statement of Cypher code:

```
MATCH  p = (n:Country {country_code: "QA"}) -[COUNTRY]- (as:AS)
-- (o:Organization)  RETURN  DISTINCT n.name,as.asn,o.name
```

[4] https://www.ihr.live/

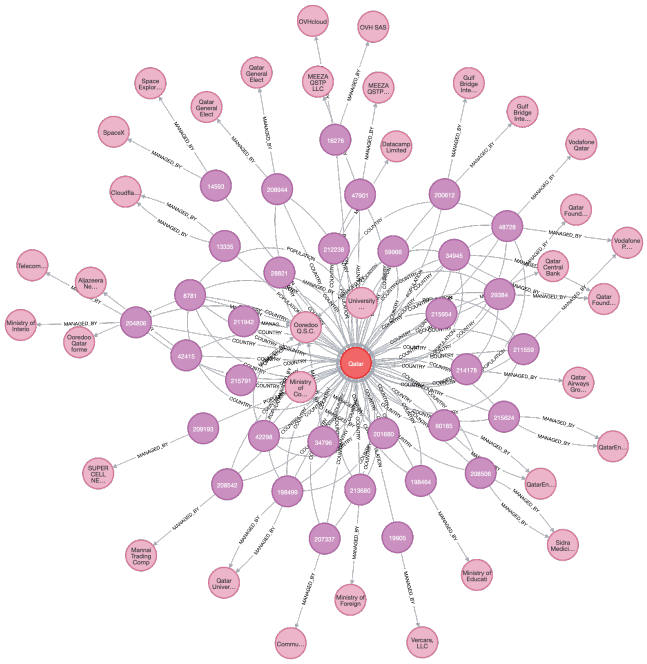

Figure 2: For the country of Qatar, all the ASs active in that country and which operator controls that AS.

7 Passive network measurement

The discussion of measurement to this point has focused on active probing. Passive measurement involves capturing data flowing over the network in order to derive results. The most obvious example of a passive measurement is a tap attached to an Internet link to monitor traffic. For instance, CAIDA operates such a probe[5]. However, this method presents challenges, including managing the enormous volume of data generated and addressing significant privacy concerns depending on the type of data collected and stored.

Another type of passive collection uses a *network telescope*, which consists of a block of Internet-connected IP addresses that host no users or services.[6]. A telescope captures primarily malicious traffic, as legitimate traffic should not reach these addresses. The observed traffic includes scans (searching for vulnerable services or exploitable features), direct attacks, and *backscatter* from randomly spoofed denial-of-service DOS attacks. In particular, if perpetrators spoof the source address of attack packets, the targeted machine may respond to these spoofed addresses, some of which likely belong to the telescope's address block. Consequently, the telescope's data can reveal which systems are under

[5] https://www.caida.org/catalog/datasets/passive_100g_dataset/
[6] https://www.caida.org/projects/stardust/docs/networktelescope/

attack.

8 The challenges of interdisciplinary research

This paper seeks to foster collaboration across research disciplines, such as between technical and policy fields. However, we recognize significant career-related barriers to such interdisciplinary work. For pre-tenure academics, publishing or collaborating outside their primary field often does not strengthen their tenure case and may even weaken it. In Computer Science, for example, presenting at a conference like TPRC is met with skepticism during tenure reviews, as TPRC papers are not peer-reviewed and the venue is unfamiliar to many evaluators. This challenge likely affects most disciplines discussed in this paper. Anecdotally, many computer science researchers engaged in interdisciplinary work are senior academics who, having secured tenure, enjoy greater flexibility to pursue the research they find meaningful.

9 For further information

- Introduction to programming the Ark probes: Getting started with Ark
  This page has pointers to further documentation.

- Introduction to Internet Yellow Pages: IYP Tutorial

References

[1] M Luckie, S Hariprasad, R Sommese, B Jones, K Keys, R Mok, and k claffy. An Integrated Active Measurement Programming Environment. In *Passive and Active Measurement Conference (PAM)*, December 2024.

A Appendix A: A more complicated example

This program, discussed in section 4 takes a DNS name as an argument, does a DNS lookup from each probe to find the address to which that name resolves at that location, then does a ping to that address from the probe, and finally writes out the data as a JSON file so it can be further processed.

Again, this is the complete program.

```
import sys
import json
from collections import defaultdict
import time
from datetime import timedelta
```

```
from scamper import *

if len(sys.argv) != 2:
  print("usage: shortest-ping.py $Target name")
  sys.exit(-1)
host = sys.argv[1]


mux = "/run/ark/mux"
ctrl = ScamperCtrl(mux=mux)
ctrl.add_vps(ctrl.vps())
ctrl.do_dns(host, inst=ctrl.instances())

targets = defaultdict(set)
for o in ctrl.responses():
    success = False
    #print (o)
    #country = o.inst.cc
    mon = str(o.inst.name).split('.')[0]

    for r in o.ans():
        #print (r, str(r))
        for lines in str(r).split('\n'):
            if lines.strip('\n').split(' ')[-2] == 'A':
                success = True
                IPr = lines.strip('\n').split(' ')[-1]
                IPn = str(IPr)
                #IPn = r.addr
                targets[mon].add(IPn)
                #print(mon, IPn)
    if success == False:
        print ('No answer from', mon)

count = 0
for inst in ctrl.instances():
    mon = str(inst.name).split('.')[0]
    if mon in targets.keys():
        #print (mon)
        for IPn in targets[mon]:
            #print(inst.name, IPn)
            ctrl.do_ping(IPn,inst = inst, wait_timeout=timedelta(seconds=3))
            break
        count += 1
    #if count > 10:
        #break
```

```
results = []

for o in ctrl.responses(timeout=timedelta(seconds=10)):
    if o.min_rtt is not None:
        vp = o.inst
        rtt = o.min_rtt
        lat = vp.loc[0]
        long = vp.loc[1]

        results.append([vp.name,lat, long,round(rtt.total_seconds()*1000,2),vp.cc, vp.st])


jfile = f'ping-{host}.json'
with open(jfile, 'w') as outfile:
    json.dump(results,outfile)
    outfile.write('\n')
```

Country	ASN	Operator
Qatar	47901	MEEZA QSTP LLC
Qatar	198499	Qatar University
Qatar	48728	Vodafone Qatar
Qatar	29384	Qatar Foundation
Qatar	16276	OVHcloud
Qatar	215624	QatarEnergy LNG PQSC
Qatar	208944	Qatar General Electricity and Water Corp.
Qatar	8781	Ooredoo Qatar formerly Qatar Telecom
Qatar	14593	SpaceX
Qatar	29384	Qatar Foundation for Education, Science and Community Development
Qatar	34945	Qatar Foundation for Education, Science and Community Development
Qatar	200612	Gulf Bridge International
Qatar	212238	Datacamp Limited
Qatar	208506	Sidra Medicine
Qatar	204806	Ministry of Interior Qatar
Qatar	60185	QatarEnergy
Qatar	201680	Ministry of Communication and Information Technology - Qatar
Qatar	13335	Cloudflare, Inc.
Qatar	42415	Aljazeera Media Network Corporation
Qatar	200612	Gulf Bridge International Inc.
Qatar	59966	Ooredoo Q.S.C.
Qatar	211942	Ooredoo Q.S.C.
Qatar	215791	Ooredoo Q.S.C.
Qatar	42298	Ooredoo Q.S.C.
Qatar	34796	Ooredoo Q.S.C.
Qatar	8781	Ooredoo Q.S.C.
Qatar	198464	Ministry of Education and Higher Education
Qatar	209193	SUPER CELL NETWORK FOR INTERNET SERVICES LTD
Qatar	16276	OVH SAS
Qatar	28821	University of Doha for Science and Technology
Qatar	208542	Mannai Trading Company LLC
Qatar	213680	Ministry of Foreign Affairs of The State of Qatar
Qatar	214178	Qatar Airways Group (Q.C.S.C) PJSC
Qatar	207337	Communications Regulatory Authority
Qatar	208944	Qatar General Electricty and Water Corporation
Qatar	14593	Space Exploration Technologies Corporation
Qatar	19905	Vercara, LLC
Qatar	47901	MEEZA QSTP-LLC
Qatar	204806	Telecommunications Department
Qatar	215954	Qatar Central Bank
Qatar	211559	Vodafone Qatar P.Q.S.C
Qatar	48728	Vodafone Qatar P.Q.S.C

Table 1: For the country of Qatar, all the ASs active in that country and which operator controls that AS.