Predicting IPv4 Services Across All Ports

Liz Izhikevich
Stanford University

Renata Teixeira
Inria, Paris

Zakir Durumeric
Stanford University

More than 300 studies have used Internet-wide scanning

Seven Years in the Life of Hypergiants' Off-Nets

Petros Gigis
University College London

Matt Calder
Microsoft & Columbia University

Lefteris Manassakis
FORTH-ICS

George Nomikos
FORTH-ICS & Lancaster University

Vasileios Kotronis
FORTH-ICS

Xenofontas Dimitropoulos
FORTH-ICS & University of Crete

Ethan Katz-Bassett
Columbia University

Georgios Smaragdakis
TU Delft

Understanding the Mirai Botnet

Manos Antonakakis[◇] Tim April[‡] Michael Bailey[†] Matthew Bernhard[◁] Elie Bursztein[○]
Jaime Cochran[▷] Zakir Durumeric[◁] J. Alex Halderman[◁] Luca Invernizzi[○]
Michalis Kallitsis[§] Deepak Kumar[†] Chaz Lever[◇] Zane Ma[†*] Joshua Mason[†]
Damian Menscher[○] Chad Seaman[‡] Nick Sullivan[▷] Kurt Thomas[○] Yi Zhou[†]

[‡]*Akamai Technologies* [▷]*Cloudflare* [◇]*Georgia Institute of Technology* [○]*Google*
[§]*Merit Network* [†]*University of Illinois Urbana-Champaign* [◁]*University of Michigan*

No study has analyzed the *entire* IPv4 service space

The IPv4 service search-space is too large

- Scanning all 65K ports across all 3.7 billion public IPv4 addresses takes 5.6 years using ZMap at 1 Gb/s

The IPv4 service search-space is too large

- Scanning all 65K ports across all 3.7 billion public IPv4 addresses takes 5.6 years using ZMap at 1 Gb/s

Solution:

- Studies often only scan assumed-relevant ports (e.g., 23/Telnet, 2323/Telnet)
- Service search engines only scan the most populated ports

Researchers are missing *billions* of IPv4 services

L. Izhikevich, R. Teixeira, and Z. Durumeric. LZR: Identifying unexpected Internet services. In *USENIX Security Symposium*, 2021.

Recent work has shown…

- Majority of services do not run on assigned ports
  - 97% of HTTP services do not occupy port 80
- Scanning the top 5K ports misses an estimated 1.9 billion (63%) of all services

L. Izhikevich, R. Teixeira, and Z. Durumeric. LZR: Identifying unexpected Internet services. In *USENIX Security Symposium*, 2021.

Recent work has shown…

- Majority of services do not run on assigned ports
  - 97% of HTTP services do not occupy port 80
- Scanning the top 5K ports misses an estimated 1.9 billion (63%) of all services
- Services on non-standard ports are not accurately represented by those on standard ports
  - IoT and vulnerable devices are up to 5 times more likely to inhabit non-standard ports

How does one *efficiently* find responsive services across all ports?

Service location is predictable

- Port usage is correlated

~50% of SMTP/465 servers also respond on IMAP/143

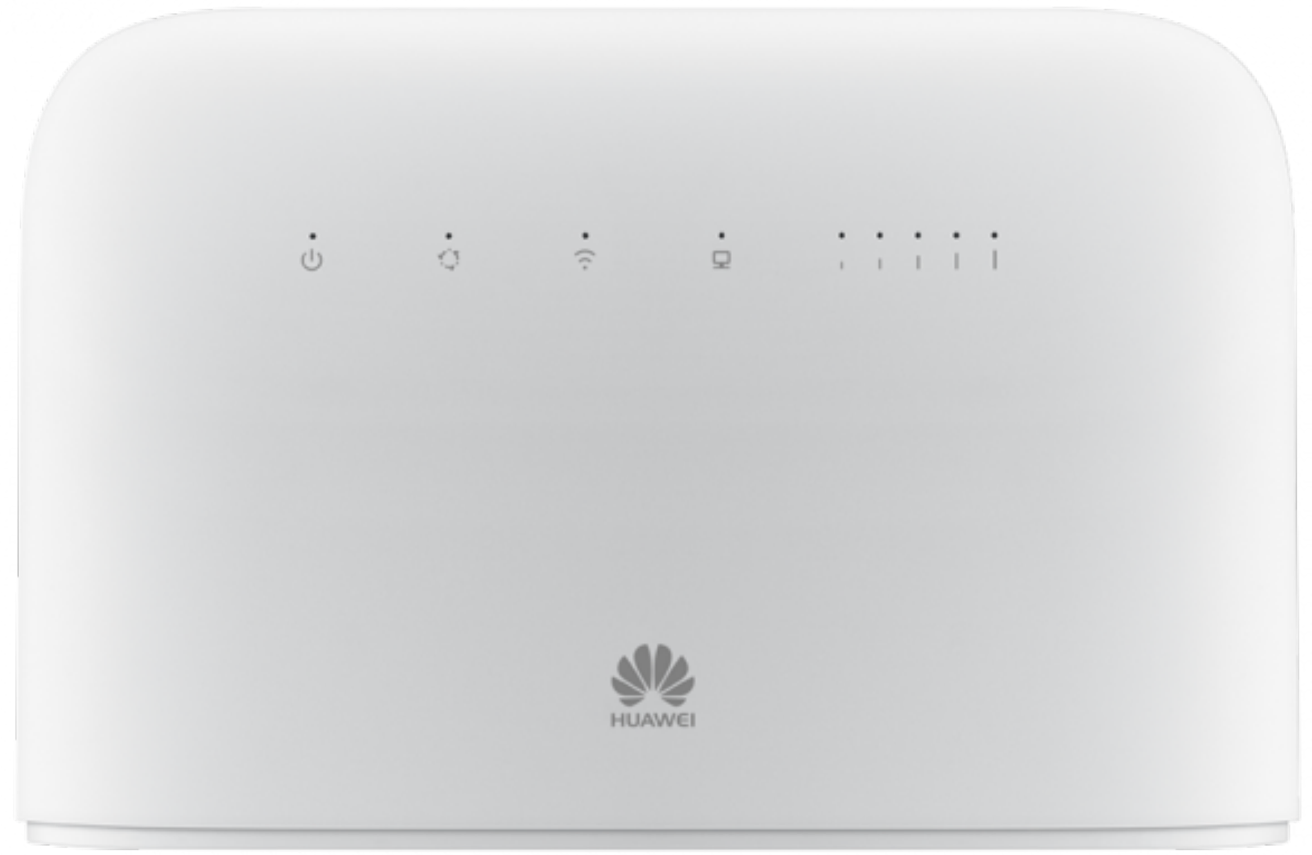
~80% of HTTP/443 also respond on HTTP/80

Service location is predictable

- Port usage is correlated
  - for every port, at least 25% of hosts responding on port A also respond on the same port B

Service location is predictable

- Different populations of hosts are more likely to run specific services
    - Fingerprinting the host-type can predict open ports



Huawei routers often serve
80/TLS and 7547/CWMP

Android Things OS often serves
8443/TLS and 8008/HTTP

Service location is predictable

- Internet services are more likely to appear together in networks

Freeboxes only appear in
networks owned by Free
(ASN 12332)

Service location is predictable

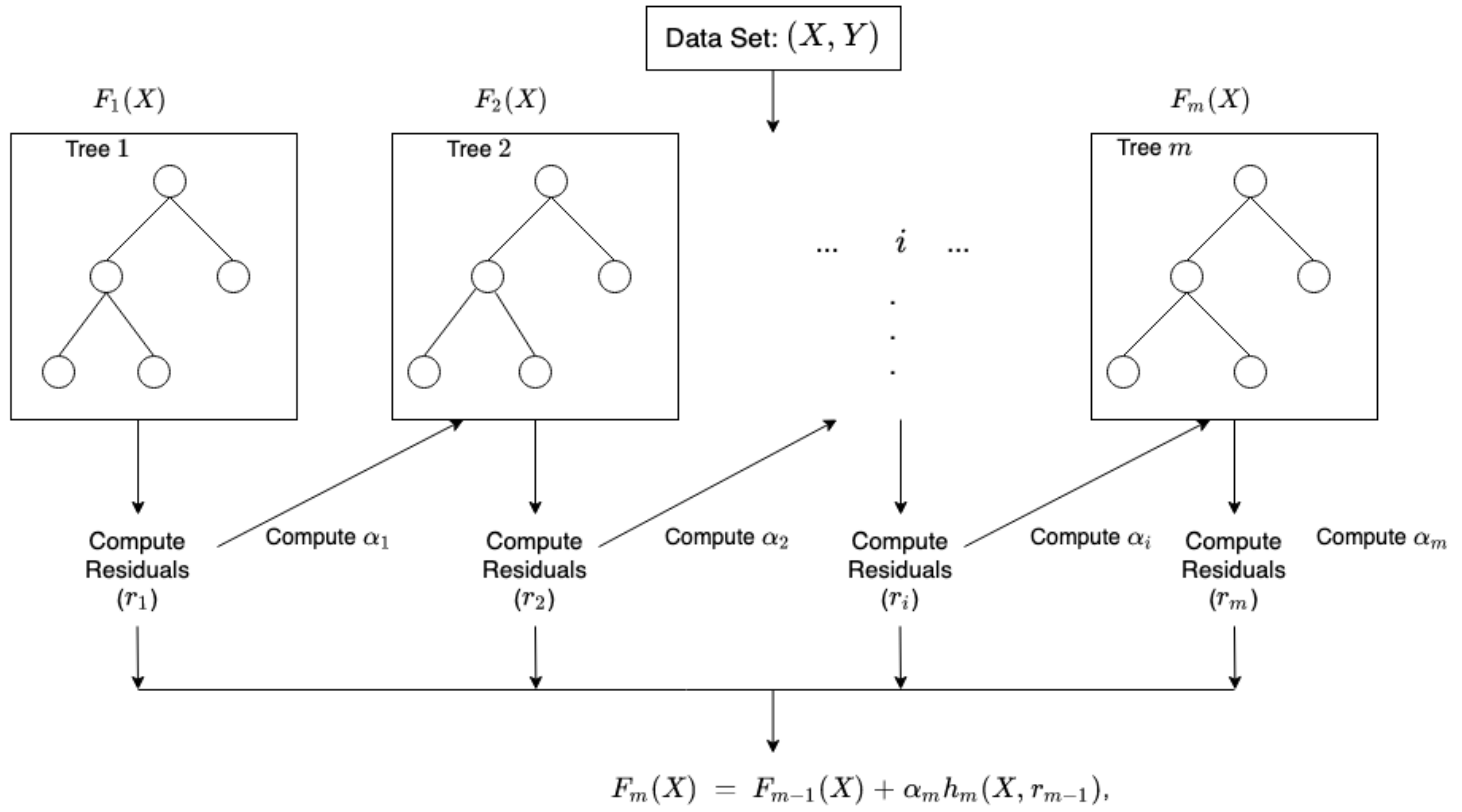
- The following categories of features predict service presence:
    - transport layer (port correlations)
    - application layer (device fingerprinting)
    - network layer (network fingerprinting)

Prior work reduces the cost of scanning by predicting responsive services

- Classifiers
- Target generation algorithm

Prior work reduces the cost of scanning by predicting responsive services

- Sarabi et al. (classifier):
  - For a list of IP addresses, train an XGBoost classifier to classify what ports a given IP address will respond on
  - Use transport, network, application layer features

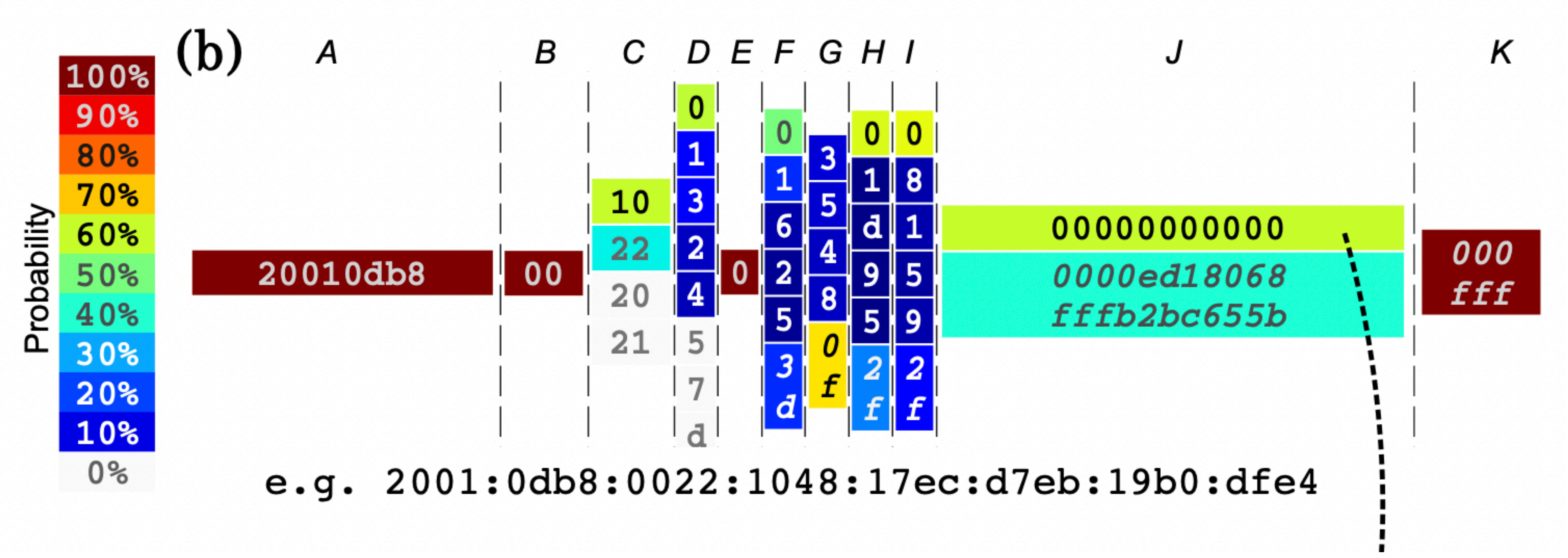


A. Sarabi, K. Jin, and M. Liu. *Smart Internet Probing: Scanning Using Adaptive Machine Learning.* 2021.

Image from https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html

Prior work reduces the cost of scanning by predicting responsive services

- Murdock et al., Foremski et al., Gasser et al., (target generation algorithms):
    - For each individual port, train a bayesian model to predict the structure of likely-responsive IP addresses
    - Only use network layer features



A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson. Target generation for Internet-wide IPv6 scanning. In *ACM Internet Measurement Conference*, 2017.

P. Foremski, D. Plonka, and A. Berger. Entropy/IP: Uncovering structure in IPv6 addresses. In *ACM Internet Measurement Conference*, 2016.

O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczyński, S. D. Strowes, L. Hendriks, and G. Carle. Clusters in the expanse: Understanding and unbiasing IPv6 hitlists. In *ACM Internet Measurement Conference*, 2018.

Existing solutions do not scale across all 65K ports

- XGBoost scanner need to be sequentially trained per port (~53 days of training)
- XGBoost scanner needs 10 million training IPs per port…which only 0.01% of ports have
- TGAs need 1,000 training IPs per port…would require one year to collect across all 65K ports using ZMap at 1Gb/s

Predicting services across all ports must…

- Train/predict in a minimum computational wall-time…because services churn quickly.
- Rely on a set of services that take minimum wall-time to scan/collect (i.e., minimum training data)

GPS: The first scalable and wall-time efficient solution for predicting IPv4 services across all ports

GPS Algorithm Overview

1. Collect a seed set (i.e., an IPv4 sample across all ports) to learn from
2. Construct a probabilistic model for service prediction
3. Use the model to predict at least one service across all likely-responsive IPv4 hosts
4. Use the model and the first found service to predict all remaining services on responsive IPv4 hosts

1. Collecting a seed set

- GPS starts with zero knowledge about Internet host -> must learn service patterns using the seed set
- The seed set consists of IPv4 services across all 65K ports

1. Collecting a seed set

- GPS starts with zero knowledge about Internet host -> must learn service patterns using the seed set
- The seed set consists of IPv4 services across all 65K ports
- The bigger the seed set, the better the predictions
- GPS can successfully predict services with just two IP samples per port (orders of magnitude smaller than prior work) across all ports

2. Identifying predictive patterns

- GPS models the interactions of the following features:
  - transport layer -> ports
  - application layer
  - network layer

Application-Layer or Network-Layer Feature
Protocol
TLS Cert: Hash
TLS Cert: Organization
TLS Cert: Subject Name
HTTP: HTML title
HTTP: Body Hash
HTTP: Server
HTTP: Header
SSH: Host Key
SSH: Banner
VNC: Desktop Name
SMTP: Banner
FTP: Banner
IMAP: Banner
POP3: Banner
CWMP: Header
CWMP: Body Hash
Telnet: Banner
PPTP: Vendor
MYSQL: Server Version
Memcached: Server Version
MSSQL: Server Version
IPMI: Banner
IP's /16 subnetwork
IP's ASN

2. Identifying predictive patterns

- GPS uses simple conditional probabilities to find the most predictive feature values

$\mathbb{P}(Port_a|Port_b)$ Transport layer correspondence

$\mathbb{P}(Port_a|(Port_b, App_{Port_b}))$ Transport and application layer correspondence

$\mathbb{P}(Port_a|(Port_b, Net_{IP}))$ Transport and layer correspondence

$\mathbb{P}(Port_a|(Port_b, App_{Port_b}, Net_{IP}))$ Transport and layer correspondence

2. Identifying predictive patterns

- Why use conditional probabilities?

(+) Simple, parallelizable calculations across all 65K ports

(+) Accurate

(+) Require minimal “training” data

2. Identifying predictive patterns

- Why use conditional probabilities?

(+) Simple, parallelizable calculations across all 65K ports

(+) Accurate

(+) Require minimal “training” data

Faster and more accurate than the XGBoost scanner

2. Identifying predictive patterns

- Why use conditional probabilities?

(+) Simple, parallelizable calculations across all 65K ports

(+) Accurate

(+) Require minimal “training” data

(-) Computationally expensive to brute force calculate the probability of all possible combinations of features

Problem: how does GPS obtain a priori information about a host?

$$\mathbb{P}(Port_a | Port_b)$$

↑

?

- The seed set only covers a small sub-set of hosts

Problem: how does GPS obtain a priori information about a host?

$$\mathbb{P}(Port_a | Port_b)$$

?

- The seed set only covers a small sub-set of hosts
- Without the model, collecting initial information about hosts is expensive as only network layer features are available

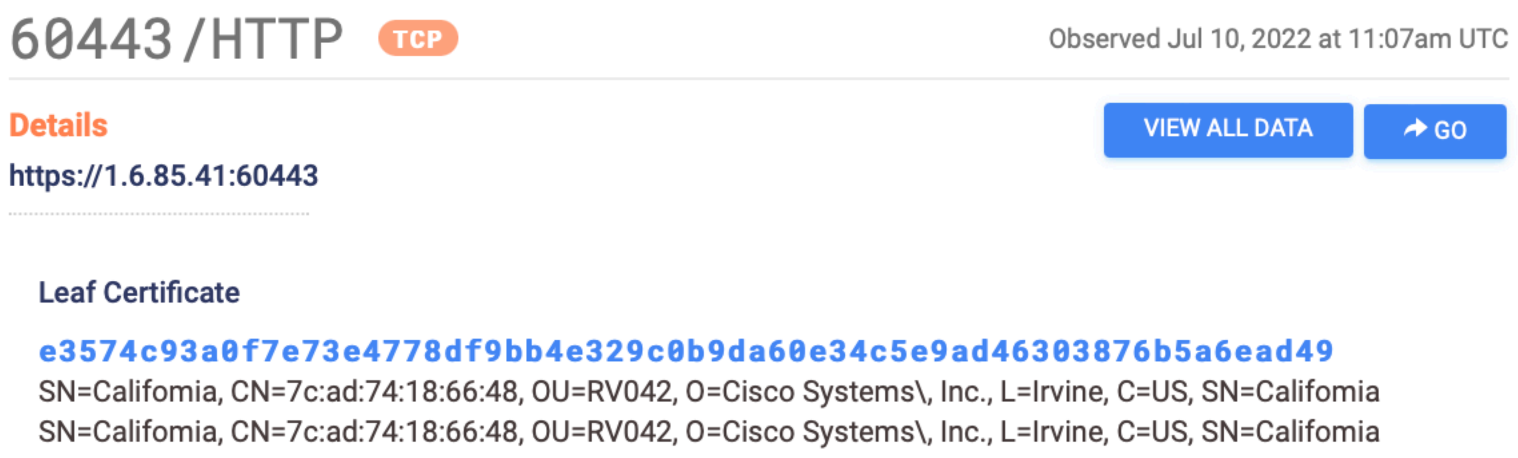
Problem: how does GPS obtain a priori information about a host?

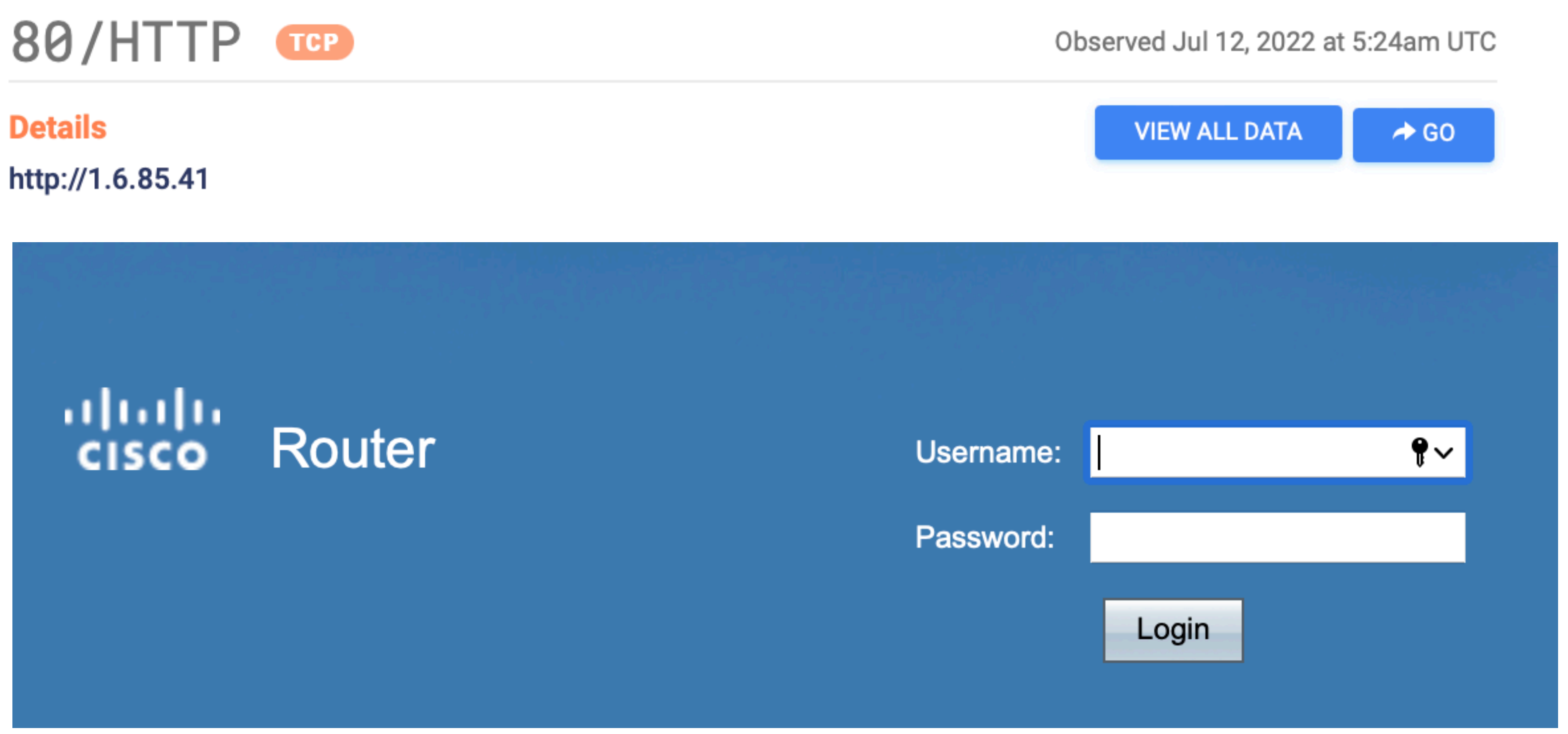
$$\mathbb{P}(Port_a | Port_b)$$

?

- The seed set only covers a small sub-set of hosts
- Without the model, collecting initial information about hosts is expensive as only network layer features are available
- Solution: collect a minimum amount of <u>most predictive information</u> about every likely-responsive host

3. Use the model to predict at least one service across all likely-responsive IPv4 hosts





P(Port 80| Port 60443) = 71%

P(Port 60443 | Port 80) = 0.2%

Port 60443’s service is more predictive of port 80’s service

3. Use the model to predict at least one service across all likely-responsive IPv4 hosts

Algorithm:

1. For all hosts that respond on only one port in the seed set, save the service's (Port #, Network_IP)

2. For all hosts that respond on more than one port in the seed set

    a. compute all four probabilistic models (e.g., P(Port_a, Port_b)) using all of the service's features

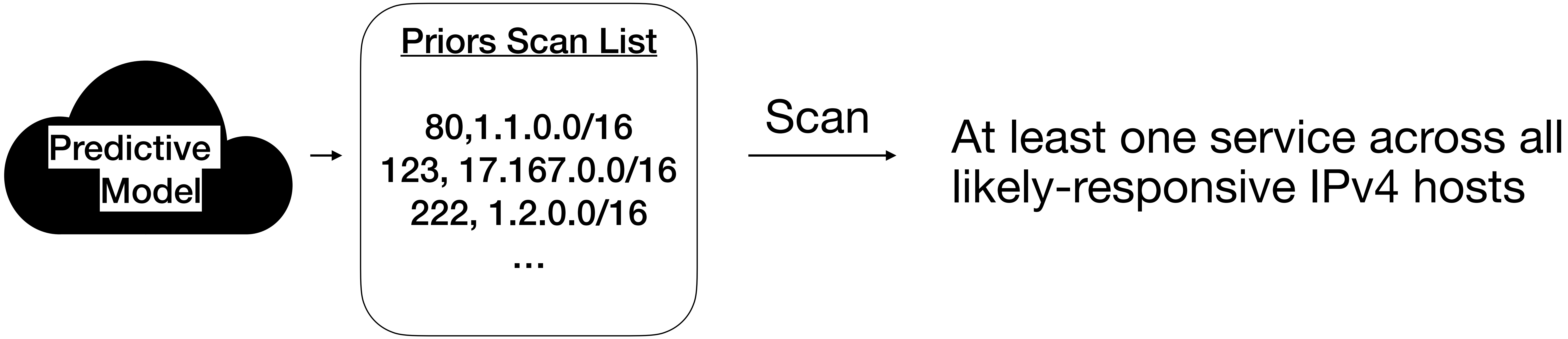
    b. Identify the Port_b that results in the maximum P(Port_a) and save the (Port #, Network_IP)

See the paper for how to determine an IP's network (e.g., ASN, /16, etc)

Priors Scan List

80,1.1.0.0/16
123, 17.167.0.0/16
222, 1.2.0.0/16
...

3. Use the model to predict at least one service across all likely-responsive IPv4 hosts



4. Use the model and first found service to predict all remaining services on responsive IPv4 hosts

Predictive Model

Priors Scan List

60443,1.6.0.0/16

...

Scan

Priors Scan Result

60443,1.6.85.41
60443,1.6.85.42
60443,1.6.85.43

...

Predictive Features List

P(port 80 | port 60443) = 71%

...

Predictions List

80,1.6.85.41
80,1.6.85.42
80,1.6.85.43

...

GPS Algorithm

1. Collect a seed set (i.e., an IPv4 sample across all ports) to learn from
2. Construct a probabilistic model for service prediction
3. Use the model to predict at least one service across all likely-responsive IPv4 hosts
4. Use the model and the first found service to predict all remaining services on responsive IPv4 hosts

GPS Algorithm

1. Collect a seed set (i.e., an IPv4 sample across all ports) to learn from
2. Construct a probabilistic model for service prediction
3. Use the model to predict at least one service across all likely-responsive IPv4 hosts
4. Use the model and the first found service to predict all remaining services on responsive IPv4 hosts

Computationally and memory expensive, but parallelizable

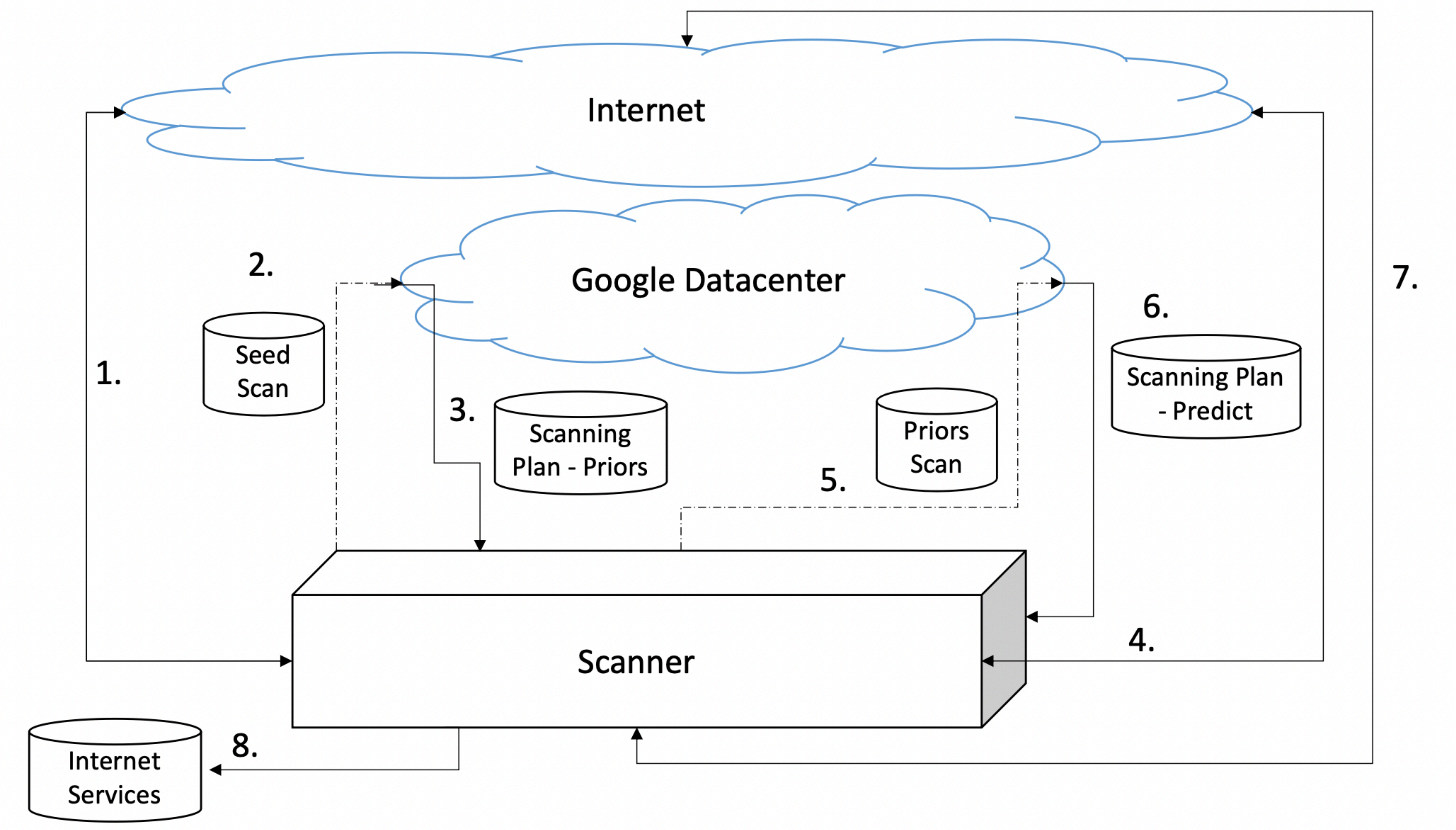
Implementing GPS with serverless compute

- Serverless computing provides an elastic and parallelizable computational environment -> minimize wall-clock time
- Google BigQuery, a serverless database platform, enables scalable analysis over petabytes of data
- Implementing GPS in a database query language makes reading, aggregating, and joining among shared fields intuitive

More details in the paper and at https://github.com/stanford-esrg/gps

Insert google big query image

GPS' implementation with serverless compute



Let’s evaluate GPS

GPS metrics for success

- GPS' objective is to maximize finding services across *all* ports

$$\text{Fraction of Services} = \frac{\#(IP, p)\ \text{Found by System}}{\#(IP, p)\ \text{in Ground Truth}}$$

Biased towards services that live on popular ports

(5% of services across all 65K ports live on only 10 ports)

GPS metrics for success

- GPS' objective is to maximize finding services across *all* ports

$$\text{Fraction of Services} = \frac{\#(IP, p) \text{ Found by System}}{\#(IP, p) \text{ in Ground Truth}}$$

$$\text{Normalized Services} = \frac{\sum_{p \in \mathcal{P}} \frac{\#IP_p \text{ Found by System}}{\#IP_p \text{ in Ground Truth}}}{|\mathcal{P}|}$$

$$\max \text{Normalized Services}(bandwidth)$$

$$bandwidth < c_1$$

Evaluating against a ground truth

- No method exists to efficiently scan 100% of IPv4 across all 65K ports
- We approximate ground truth using two datasets:
  - Censys 100% IPv4 scan across the most popular 2K ports
  - LZR 1% IPv4 scan across all 65K ports

Evaluating against a ground truth

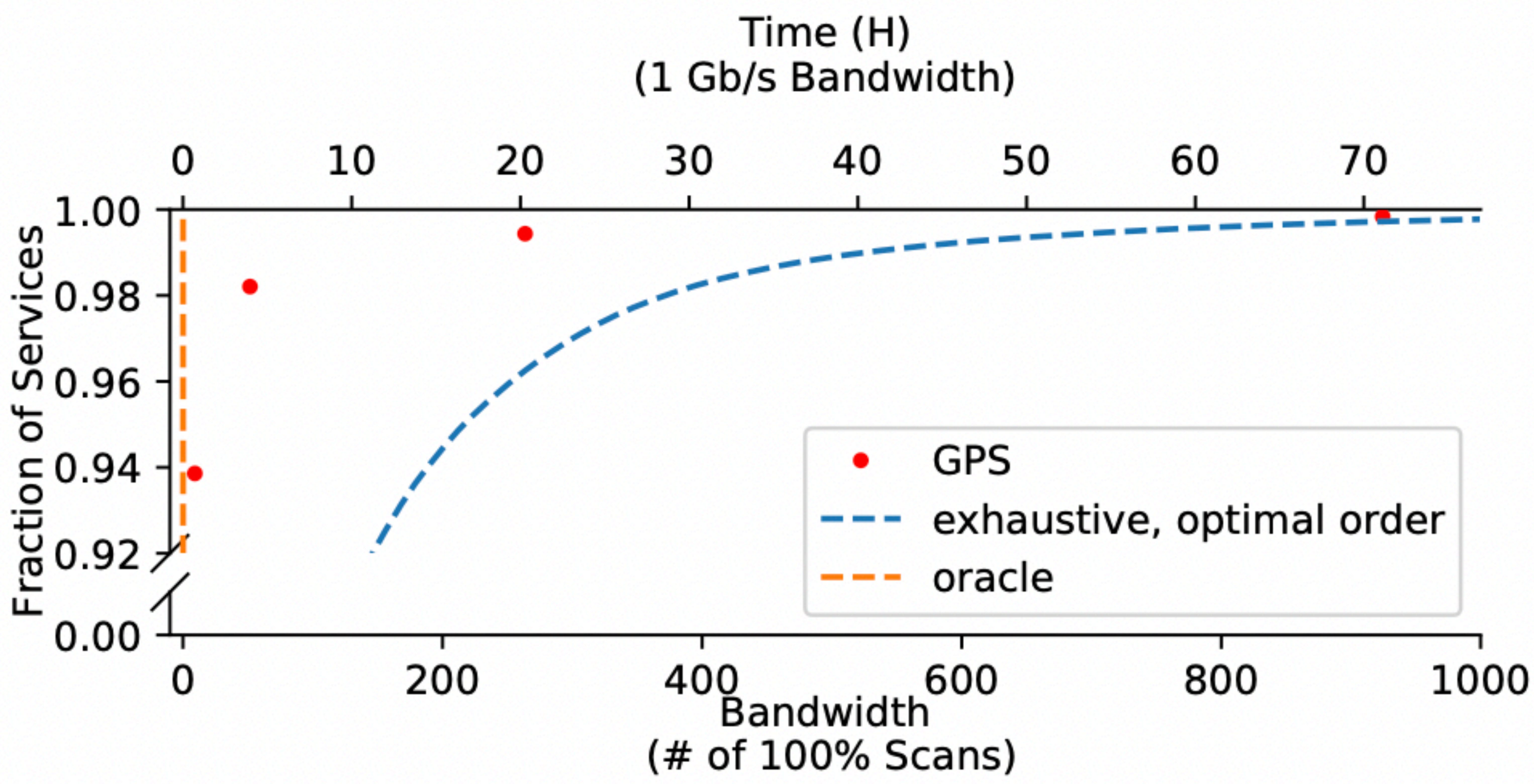
- No method exists to efficiently scan 100% of IPv4 across all 65K ports
- We approximate ground truth using two datasets:
    - Censys 100% IPv4 scan across the most popular 2K ports
    - LZR 1% IPv4 scan across all 65K ports

Creating a tighter benchmark for GPS

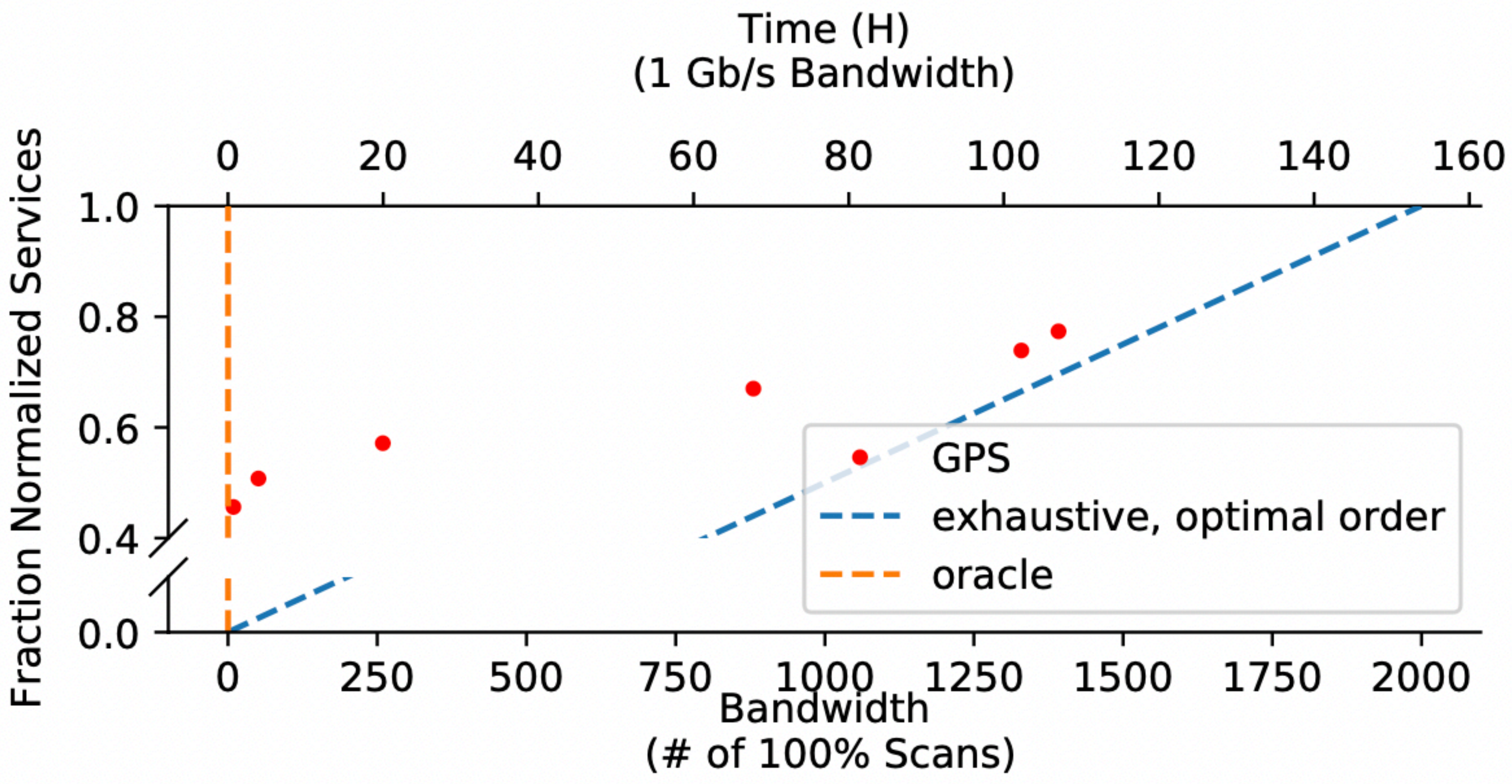
- No method exists to efficiently scan 100% of IPv4 across all 65K ports
- Evaluate against “exhaustive, optimal port-order probing”: exhaustively scanning the minimum number of ports to find the maximum fraction of services

Ports	Fraction of all services	Fraction of normalized services
80		1/65K
80, 443		2/65K
80, 443, 7457		3/65K

GPS finds 94% of all services using 21x less bandwidth than optimal port-order probing

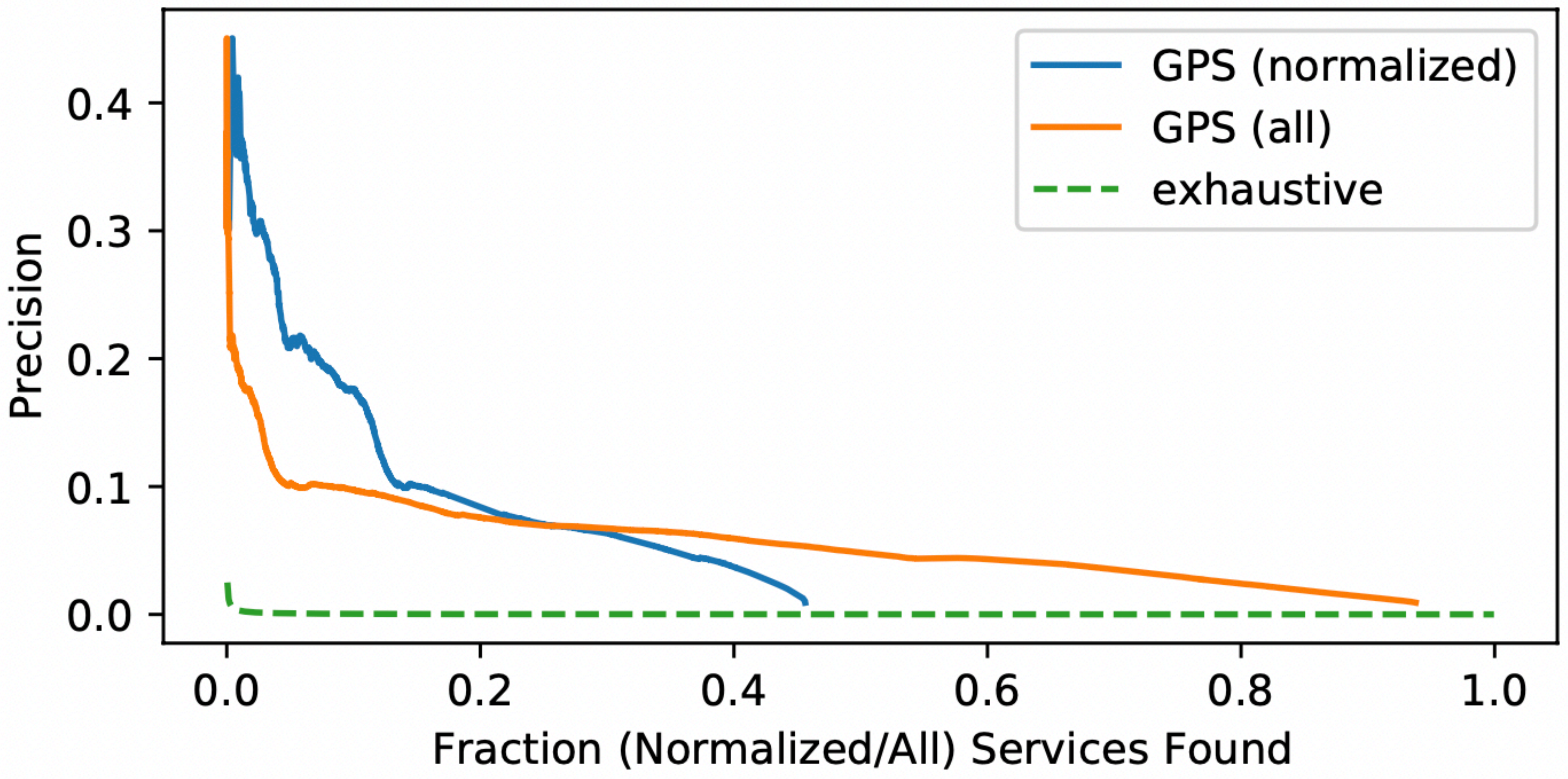


GPS finds 46% of normalized services using 100x less bandwidth than optimal port-order probing and 67% of normalized services using 50% less bandwidth



Stanford University

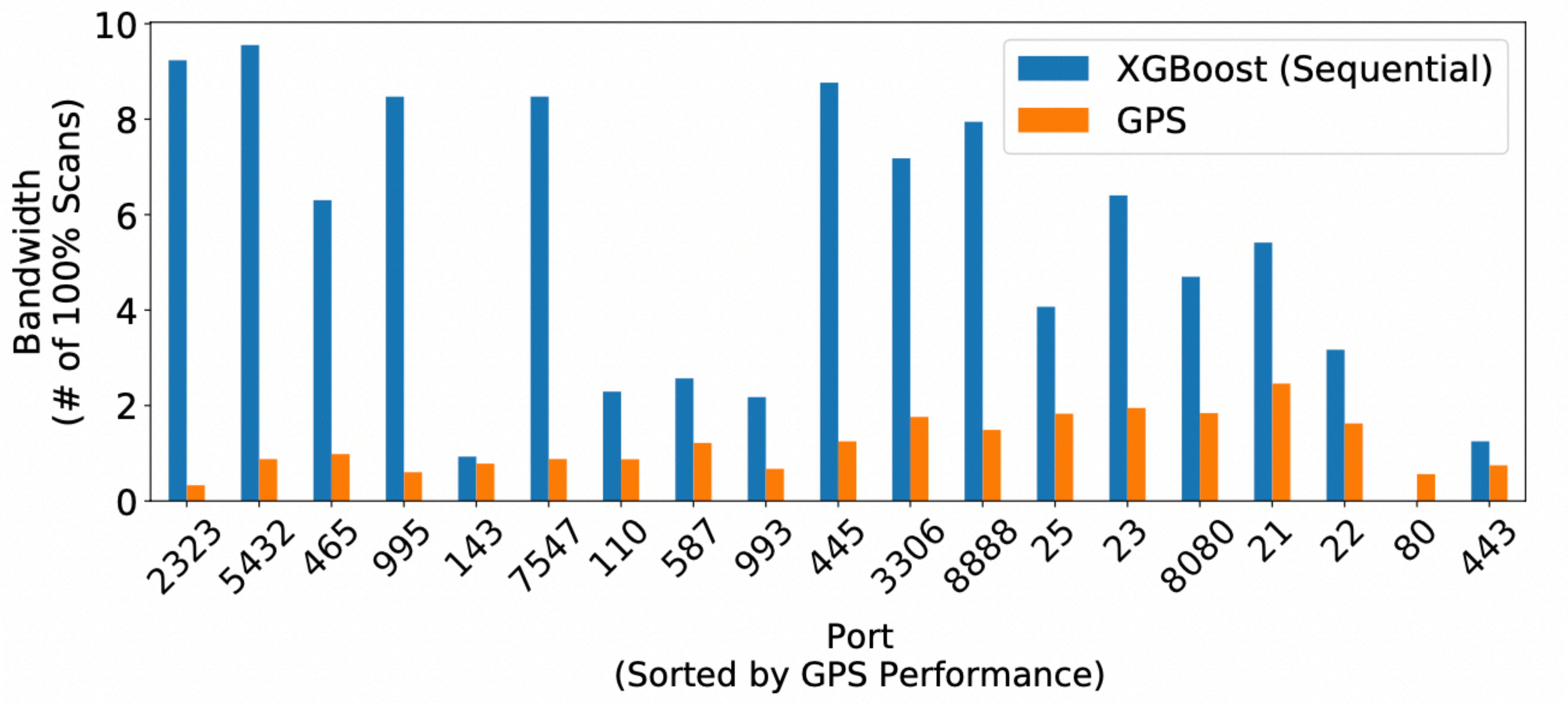
GPS finds 94% of all services and 46% of normalized services while being over 10x more precise than exhaustive probing



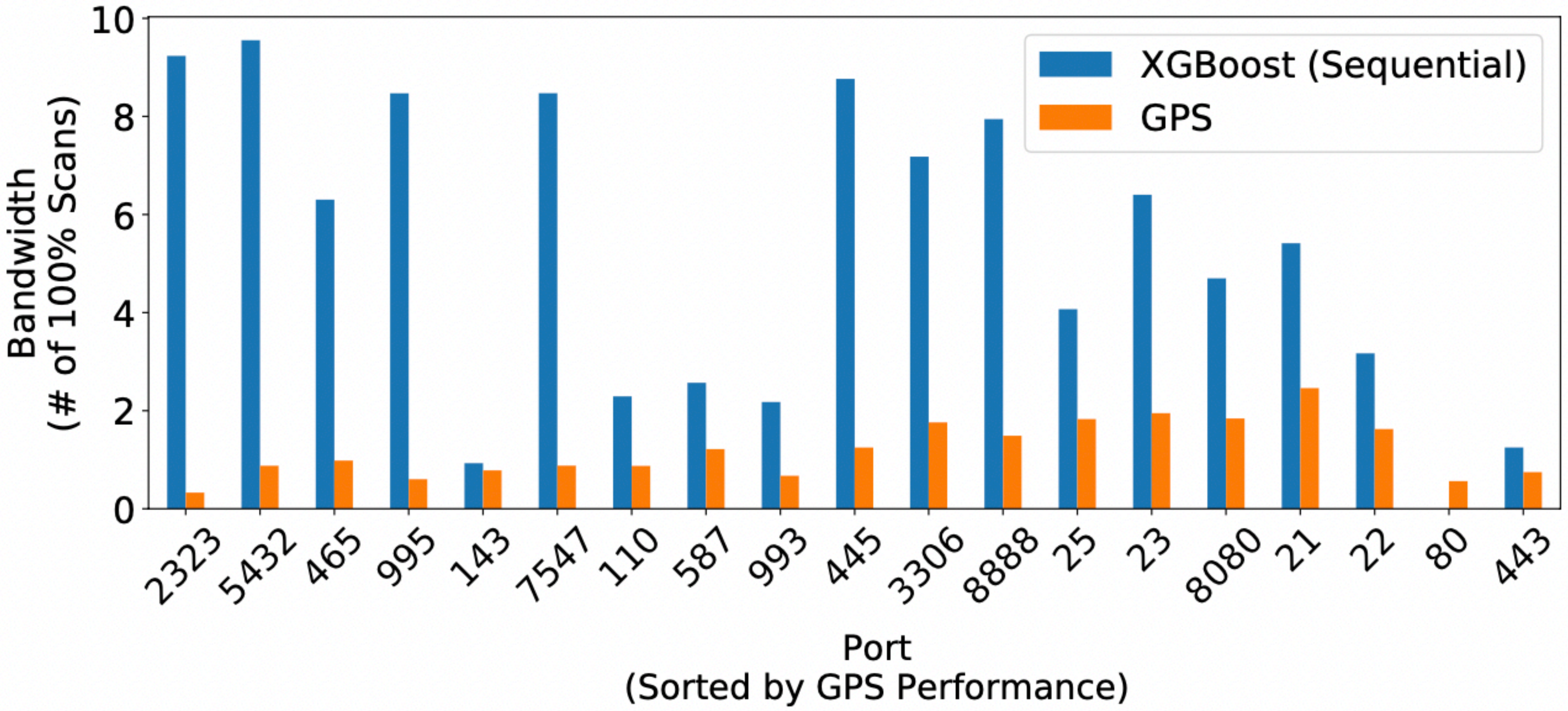
Evaluating against the XGBoost scanner

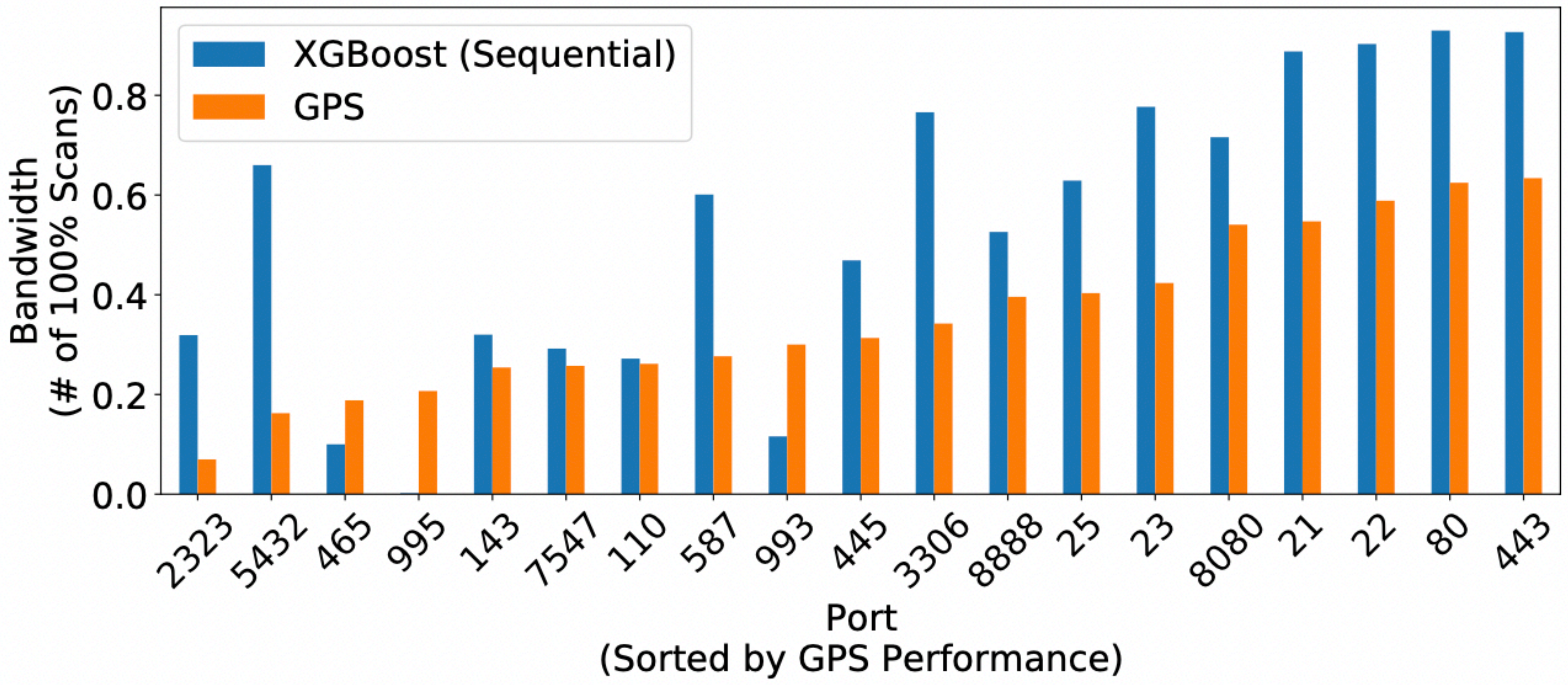
- Sarabi et al. train an XGBoost classifier to predict services on a target port using two phases:
    1. Use the XGBoost classifier to predict services on alternate ports that are considered predictive for the target port
    2. Use the output of the previous scan to help predict services on the target port

GPS saves up to 28x more bandwidth than XGBoost scanner when collecting the minimum set of predictive services

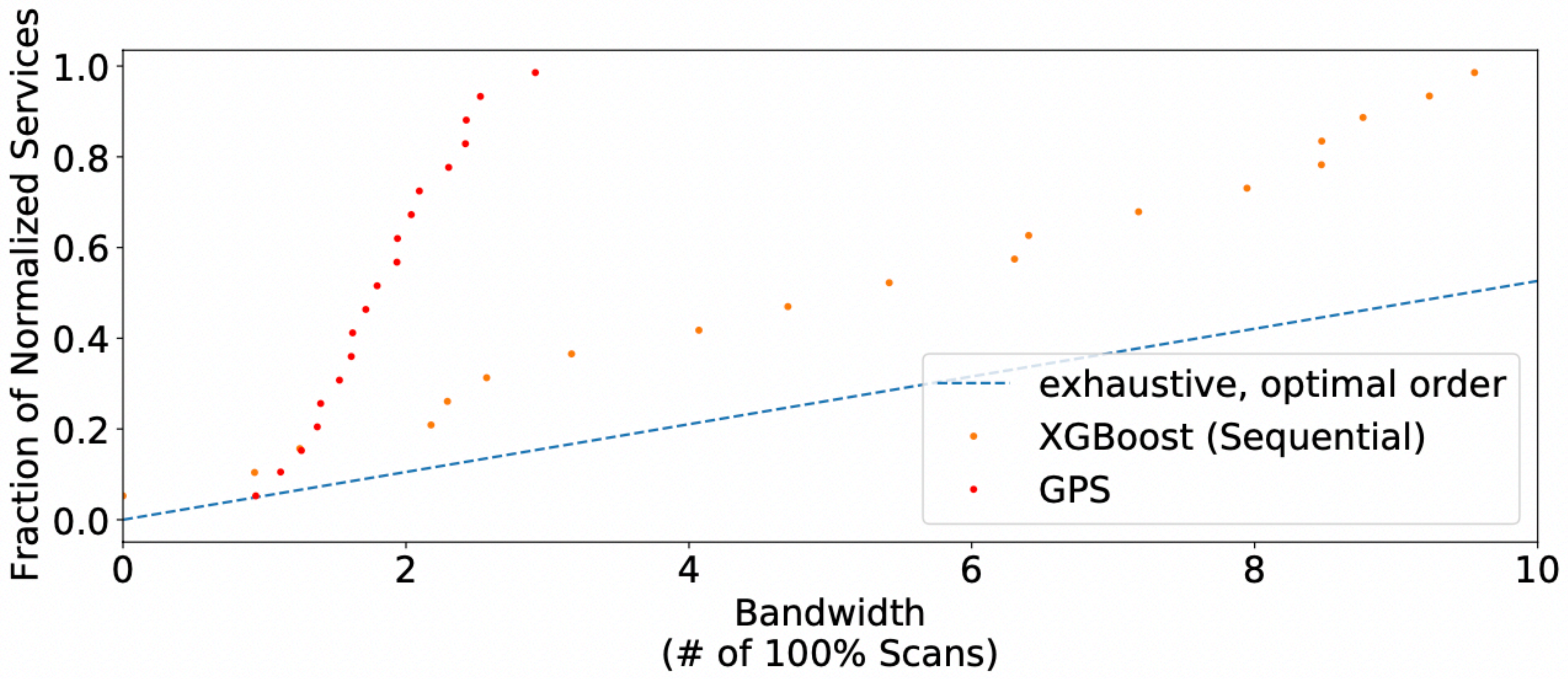


GPS saves more bandwidth than XGBoost scanner when scanning 16/19 popular ports





GPS uses 3x less bandwidth to find 98.5% of normalized services than XGBoost scanner



Computational Complexity - Time

- Using a **single core**, GPS performs predictions in **9 days and 9 hours** — 5.6x faster than XGBoost scanner
- Using **serverless** computing, GPS performs predictions in **13 minutes** — 10000x faster than XGBoost scanner

Computational Complexity - Time

- Using a **single core**, GPS performs predictions in **9 days and 9 hours** — 5.6x faster than XGBoost scanner
- Using **serverless** computing, GPS performs predictions in **13 minutes** — 10000x faster than XGBoost scanner
- GPS' bottleneck is bandwidth:
    - Collecting the seed scan, if it is not available, can take days/months
    - Data transfer to/from Google BigQuery is bottlenecked by Google's limits

With an available seed scan, GPS takes a total of 9 hours to predict and scan all services

Computational Complexity - Space

- Required memory is dependent upon:
  - Size of seed scan (e.g., filtered LZR 1% IPv4 = 4GB)
  - Number of features to extract
  - The conditional probability algorithm (can create a memory footprint 50 times larger than seed scan size)

Computational Complexity - Space

- Required memory is dependent upon:
  - Size of seed scan (e.g., filtered LZR 1% IPv4 = 4GB)
  - Number of features to extract
  - The conditional probability algorithm (can create a memory footprint 50 times larger than seed scan size)
- Final list of 28 billion predicted services is 547GB (~100x greater than the initial seed scan file)

Most predictive features

Feature	Normalized Services	Services
(Port, $\text{Port}_{\text{Protocol}}$)	18.7%	2.0%
Port	14.1%	2.0%
(Port, $\text{Port}_{\text{HTTP Header}}$)	9.7%	2.0%
(Port, Port_{ASN}, $\text{Port}_{\text{HTTP-Body-Hash}}$)	7.7%	2.0%
(Port, $\text{Port}_{\text{HTTP-Body-Hash}}$)	6.1%	2.0%

Limitations for predictive Internet scanning

- IPv6 search space
    - GPS relies on exhaustively scanning sub-networks to find the first service
    - GPS can be used to predict additional services on the same IPv6 address when one is already known

Limitations for predictive Internet scanning

- Some patterns will never be predictive
  - Random host configuration
  - FRITZ!Box : “for security reasons, FRITZ!Box sets up a random TCP port for HTTPS when internet access via HTTPS is enabled”
  - Routers port-forward services through random ports

Conclusion

- GPS is a scanning system that predicts IPv4 services across all ports and finds *billions* of previously-hidden services

Conclusion

- GPS is a scanning system that predicts IPv4 services across all ports and finds *billions* of previously-hidden services
- To predict services, GPS parallelizes conditional probability calculations

Conclusion

- GPS is a scanning system that predicts IPv4 services across all ports and finds *billions* of previously-hidden services
- To predict services, GPS parallelizes conditional probability calculations
- GPS finds 94% of services using 21x less bandwidth than exhaustive scanning

Conclusion

- GPS is a scanning system that predicts IPv4 services across all ports and finds *billions* of previously-hidden services
- To predict services, GPS parallelizes conditional probability calculations
- GPS finds 94% of services using 21x less bandwidth than exhaustive scanning
- GPS calculates all predictions in 13 minutes

Conclusion

- GPS is a scanning system that predicts IPv4 services across all ports and finds *billions* of previously-hidden services
- To predict services, GPS parallelizes conditional probability calculations
- GPS finds 94% of services using 21x less bandwidth than exhaustive scanning
- GPS calculates all predictions in 13 minutes
- GPS is open source: https://github.com/stanford-esrg/gps

Conclusion

Questions?

- GPS is a scanning system that predicts IPv4 services across all ports and finds *billions* of previously-hidden services
- To predict services, GPS parallelizes conditional probability calculations
- GPS finds 94% of services using 21x less bandwidth than exhaustive scanning
- GPS calculates all predictions in 13 minutes
- GPS is open source: https://github.com/stanford-esrg/gps