Predicting IPv4 Services Across All Ports

Liz Izhikevich
Stanford University

Renata Teixeira
Inria, Paris

Zakir Durumeric
Stanford University

ABSTRACT

Internet-wide scanning is commonly used to understand the topology and security of the Internet. However, IPv4 Internet scans have been limited to scanning only a subset of services—exhaustively scanning all IPv4 services is too costly and no existing bandwidth-saving frameworks are designed to scan IPv4 addresses across all ports. In this work we introduce GPS, a system that efficiently discovers Internet services across all ports. GPS runs a predictive framework that learns from extremely small sample sizes and is highly parallelizable, allowing it to quickly find patterns between services across all 65K ports and a myriad of features. GPS computes service predictions in 13 minutes (four orders of magnitude faster than prior work) and finds 92.5% of services across *all* ports with 131× less bandwidth, and 204× more precision, compared to exhaustive scanning. GPS is the first work to show that, given at least two responsive IP addresses on a port to train from, predicting the majority of services across all ports is possible and practical.

CCS CONCEPTS

• **Networks** → Network structure; • **Computing methodologies** → *Model development and analysis*; • **Information systems** → *Data mining*;

KEYWORDS

Internet Scanning, Prediction, IPv4, Measurement

ACM Reference Format:
Liz Izhikevich, Renata Teixeira, and Zakir Durumeric. 2022. Predicting IPv4 Services Across All Ports. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3544216.3544249

1 INTRODUCTION

Internet-wide scanning allows researchers and network operators to understand how the Internet works *in practice*, and has been used to study network topology [16, 38], operator decisions [20, 28] and security vulnerabilities [15, 17]. Internet-wide scanning works by initiating network connections with services on a set of given ports. Unfortunately, no study has been able to analyze the entire IPv4 service space across all ports, as scanning all 65K ports across all 3.7 billion IPv4 addresses would require 5.6 years using ZMap [21] at 1 Gbps—a scanning rate that prevents flooding destination networks. As a result, Internet-wide studies often scan only a relevant subset of assigned ports (e.g., 23/TELNET and 2323/TELNET), and popular Internet-service "search engines" like Censys [19] and Shodan [37] resort to scanning only the most populated ports.

However, recent work has shown that the majority of Internet services do not run on assigned ports. Scanning only port 80 misses 97% (1.8 billion) of all IPv4 HTTP services and scanning only port 23 misses 95% (6.8 million) of Telnet services [25]. Services occupy a long tail of nonstandard ports [25]: Internet service search engines that scan the five thousand most populated ports (e.g., Censys) miss the majority—an estimated 1.9 *billion* (63%)—of Internet services. To complicate matters, the services on non-standard ports are not accurately represented by those on standardized ports: IoT and security-critical devices are up to five times more likely to inhabit the 1.9 billion services on non-standard ports [25]. Unfortunately, botnets already target vulnerable services on non-standard ports [13]. Further, both researchers and network operators often cannot rely on sub-sampling when needing to find a "needle in the haystack" in the long tail of Internet services. For example, Marczak et al. rely on complete Internet-wide scans of select ports to identify spyware infrastructure [31] and clients of cyber espionage [32], which occupy only a few hundred compromised servers. For researchers and operators to secure the entire Internet, it is imperative to find services across all IP addresses *and* ports.

In this work, we introduce GPS, an intelligent scanning system that scalably and efficiently finds services across all IPv4 addresses and ports. GPS assumes no prior knowledge and starts by collecting an initial set of "seed" data. It uses a myriad of application, transport, and network layer features to probabilistically model service presence. GPS uses the model to construct a two-phased scanning approach that: (1) finds at least one service on all hosts, and (2) uses the discovered service to find remaining services on the same host. GPS's key contribution is a parallelizable and accurate predictive algorithm—relying on calculating simple conditional probabilities between features and services—that can find patterns and predict new services across all ports in parallel with minimal training data.

GPS finds 92% of services across all 65K ports (with greater than two responsive IP addresses) using 131 times less bandwidth and 204 times higher precision than exhaustive scanning. We show that as services become harder to predict, the bandwidth required to find new services increases: finding 96% of services across all ports uses only 10 times less bandwidth compared to exhaustively scanning. When scanning only popular ports, GPS's algorithm uses up to 28 times less bandwidth than the state-of-the-art ML-based solution [36]. On a single CPU core, GPS computes predictions five times faster than if prior work was extended to predict services across all ports. When using elastic cloud platforms, GPS's parallelizeable algorithm computes predictions in 13 minutes—four orders of magnitude faster than extending prior work.

GPS is the first practical system that finds the majority of services across all ports on the IPv4 address space. We are releasing

SIGCOMM '22, August 22–26, 2022, Amsterdam, Netherlands

ACM ISBN 978-1-4503-9420-8/22/08…$15.00
https://doi.org/10.1145/3544216.3544249

GPS under the Apache 2.0 License so that individual researchers, network operators, and security companies can reduce the needed bandwidth to find, study, and secure the majority of exposed—and perhaps already exploited—services on the Internet.

2 PRIOR WORK

To reduce the cost of scanning and identify additional services, prior work has proposed several solutions to predict which IP addresses to scan on a given port. There exist two drawbacks across prior work: (1) no system is designed—nor scales—to predict services across *all* 65K ports, and (2) existing solutions require a prohibitive amount of training data, which would take years to collect due to the sparse search space.

Classifiers. Sarabi et al. [36] approach intelligent Internet-wide scanning as a classification problem—in which each port value is a class—and use an XGBoost classifier [18] to predict whether an IP will respond on one of 20 popular ports. Their work finds that the strongest predictor of a service is the presence of other services on the same host. To find patterns across common ports, their system sequentially trains a model per port and uses the output of each model to train the next. Unfortunately, their method does not scale to all 65K ports for two reasons. First, their model requires at least 10 million services to train on per port, which is not available across 99.99% of ports (many ports simply do not have that number of services present). Second, an individual model must be trained for each port, making the computation time prohibitively expensive: the training and prediction process per port requires roughly 70 seconds on an NVIDIA GeForce GTX 1080 Ti GPU, and requires roughly 53 days of computation across all 65K ports. This computation cannot be parallelized, because the training of the models is sequential. We compare the performance of XGBoost scanner to GPS in Section 6.4, and show that GPS is more accurate than XGBoost on average, is capable of predicting services across all ports, and takes four orders of magnitude less time to compute service predictions.

Target generation algorithms. Previous work [22, 23, 35] has predicted services on IPv6 addresses using *Target Generation Algorithms* (TGAs). TGAs learn the structure of known IP addresses and predict similarly structured addresses that are likely to run services. However, TGAs are unable to find patterns across all 65K ports and require a new model to be built and trained for each port. TGAs also rely solely on subnetwork correlations, which we show are orders of magnitude less predictive of hosts on uncommon ports (Section 4). Most importantly, TGAs are not computationally scalable: obtaining the minimum number of IP addresses required to effectively train the model (i.e, 1,000 IPs [22]) across 90% of ports would require randomly probing at least 25% of the address space per port—due to the sparsity of most uncommon services—requiring over one year to collect using ZMap [21] at 1Gbps.

We verify whether TGAs are capable of predicting services on IPv4 addresses across densely-populated ports by modifying Entropy/IP [22] and EIP [23] to predict IPv4 addresses (predicting one IPv4 octet at a time instead of one IPv6 nibble). We use 1,000 randomly sub-sampled addresses from Censys' Universal Internet Dataset containing 100% IPv4 scans across 2K ports to train a model for each port. Each model predicts 1M candidate addresses per port (an order of magnitude more addresses than the number of IPs that respond across 90% of ports). Combining all candidate addresses, Entropy/IP and EIP are able to find only 19% of services in the Censys dataset.

Recommendation Systems. Proprietary recommendation systems have been successful at recommending millions of items to millions of users [24, 30, 33]. We apply a popular open-sourced hybrid recommender [26] to recommend candidate ports to IP addresses and find it to be unsuccessful at predicting services (Appendix A).

3 GPS OBJECTIVE

GPS's objective is to maximize finding services across *all* ports. This differs from prior Internet scanning systems [36] that maximize the total "fraction of services" found (Equation 1): the number of services found by the system relative to the number of IPs that are found in a "ground truth" set (i.e., a baseline for the true number of services on port p, obtained by a 100% scan). This metric is biased towards discovering services on popular ports where services are more dense (e.g., 5% of all services across all 65K ports live on the top 10 ports) and disincentives finding services on uncommon ports, which are understudied and more likely to be vulnerable [25].

$$\text{Fraction of Services} = \frac{\#(IP, p) \text{ Found by System}}{\#(IP, p) \text{ in Ground Truth}} \tag{1}$$

To address this, we introduce an additional *normalized service* metric (Equation 2) which, given port p, normalizes the weight of a service based on the number of IPs that respond on port p (i.e., IP_p) in a "ground truth" set. By doing so, discovering all services on an uncommon port holds equal weight compared to discovering all services on a popular port. Using the new metric, GPS's goal (Equation 3) is to maximize the fraction of normalized services found across the set of all ports ($|\mathcal{P}|$), while constraining the number of probes (i.e., bandwidth) by constant c_1.

$$\text{Normalized Services} = \frac{\sum_{p \in \mathcal{P}} \frac{\#IP_p \text{ Found by System}}{\#IP_p \text{ in Ground Truth}}}{|\mathcal{P}|} \tag{2}$$

$$\max \text{Normalized Services}(bandwidth) \quad bandwidth < c_1 \tag{3}$$

By constraining bandwidth, GPS is optimized to scan only the most predictable services in order to maximize the total fraction of normalized services found. The more bandwidth available, the more services GPS finds.

There are two additional constraints when building a deployable system:

(1) **Computationally scalable:** Predicting services across all ports and IP addresses encompasses a large search space, but requires a solution that operates in a constrained amount of wall-time. This is especially important as Internet services are constantly changing [34]. For example, we conduct a scan of the same 0.1% of the IPv4 address space across 65K ports on June 17, 2021 and June 27, 2021. Within the 10-day period, 15% of normalized services and 9% of all services disappear. Thus, if predicting services takes too long, the initial set of services that the model uses to learn patterns from,

and the model's predictions, will likely become obsolete.

(2) **No prior knowledge:** Intelligent Internet scanning is not a classic prediction task in which informative features are initially available for training. Predictive features reside on IP addresses and ports that are not initially known. Since GPS starts with no prior information about the location of Internet services, it must devise an efficient scanning strategy that gradually discovers information about hosts, which can later be used to predict other services.

Ethics. Any open source scanning technique can be used both by researchers to monitor and secure Internet services, and by attackers to uncover vulnerabilities. We design GPS to be easily blocked by network operators (Section 5.5), which disincentives attackers from using GPS while enabling researchers to gain visibility of and secure the Internet. When executing scans and building GPS, we follow the community standards for good Internet citizenship outlined by Durumeric et al. [21]. We also emphasize that GPS reduces the traffic sent when scanning services on a large number of ports (Section 6), thereby reducing the impact on destination networks. We hope that researchers and companies both consider using GPS to reduce the total number of scan probes on the Internet.

4 IDENTIFYING PREDICTIVE FEATURES

At first glance, finding IPv4 services across 3.7 billion addresses and 65K ports—a nearly 2^{48} search space—may seem intractable. However, prior studies have surfaced several ways to predict the locations of *popular* services. We begin by investigating whether these predictive patterns can be used to predict services across all 65K ports. Our results build the foundational set of three categories of predictive features that GPS will rely on: network layer, transport layer, and application layer.

Port usage is correlated (Transport layer). Bano et al. showed that among the eight most popular ports, the presence of one port on a host can be used to predict the presence of other ports [14]. We scan all 65K ports on a 1% random IPv4 sample in March 2021 and find the same holds true across the majority of all ports: for every port, at least 25% of hosts also respond on the same second port. Thus, a service's port can be used to predict other open ports on the same host.

Different populations of hosts are more likely to run specific services (Application layer). Prior work [25, 27] shows that IoT and router vendors often manufacture particular ports to be open in order to provide network access. Thus, application-layer data (e.g., TLS certificates) that indicates manufacturer or operating system can likely be used to predict other services on the same host. Using the same scan, we manually investigate the top 3K most common HTTP header values, SSH banners, and TLS certificates across all hosts. We find that IoT devices and routers are the most popular host type across the majority of ports. Having devices with manufactured—and thus predictable—port presence dominate services across all ports encourages the pursuit of predicting services on uncommon ports. Application layer features that identify a host's manufacturer (e.g., the organization, subject name or issuer of a TLS certificate, SSH Banner, PPTP vendor, etc.), operating system (e.g., HTTP Server, SSH banner, CWMP Header, MySQL server version, etc), purpose (e.g., HTTP HTML title, VNC desktop name, etc), and owner (e.g., SSH key, TLS certificate, etc.) can predict service presence.

Internet services are more likely to appear together in networks (Network layer). Murdock et al. and Foremski et al. found that hosts in a given network are more likely to have the same ports open [22, 35]. Using Censys' universal data set comprising of 100% IPv4 scans of over 2K ports [11], we find that 81% of all services appear at least twice on the same port within the same /16 subnetwork. This result indicates that the network is predictive of service presence. However, as ports become less popular, the probability of finding two services responding on the same port in the same /16 subnetwork becomes as low as 0.02%. Thus, while network patterns are effective at finding hosts with a popular port open, discovering services on uncommon ports cannot be done by solely relying on network patterns.

In Section 5.2, GPS uses transport-layer, application-layer, and network-layer features to predict service presence.

5 SYSTEM ARCHITECTURE

In this section, we introduce GPS, a system that efficiently finds Internet services across all ports. GPS assumes no prior knowledge and uses a four phase process to bootstrap itself and comprehensively discover services. First, through random sampled scanning, GPS collects a "seed set" of Internet hosts and services to learn from. Second, GPS creates a probabilistic model of the most predictive feature values across all services in the seed set. Third, because GPS's seed set spans only a subset of hosts, GPS uses its probabilistic model to find at least one service on each responsive IPv4 host. Finding the first service of a host is non-trivial and bandwidth expensive, but is crucial to predicting remaining services. Fourth, once GPS has discovered at least one service on each responsive host, GPS predicts remaining services.

GPS's key insight is using a simple and parallelizable computation for predicting services that is not dependent upon a large training set. While GPS's algorithm is computationally expensive, we provide an implementation that marries serverless computing's elastic resources with GPS's parallelization, speeding up GPS's wall-clock prediction time by orders of magnitude.

5.1 Building a Seed Set

GPS starts with no knowledge about Internet hosts. To learn patterns and find services, GPS must either collect, or use an available (e.g., the LZR dataset [3]) uniform random sample of IPv4 hosts each scanned across all 65K ports (i.e., "a seed set"). The size of the seed scan directly impacts prediction quality and is the primary way to ensure that GPS will learn all predictive patterns. We show in Appendix D.2 the bandwidth/coverage trade-off when determining the size of a seed-scan: larger seed sizes find more services, but use substantially more bandwidth. Importantly, unlike prior work (Section 2) that requires large sample sizes (e.g., 25% IPv4 sample) to train, GPS is able to predict services across the majority of ports with only a 0.1% IPv4 sample. GPS allows the user to specify the desired size of the seed scan as an input parameter based on their bandwidth constraint (Equation 3).

Application-Layer or Network-Layer Feature	# Unique Values in Censys Ground Truth
Protocol	56
TLS Cert: Hash	30.1M
TLS Cert: Organization	1.1M
TLS Cert: Subject Name	27.9M
HTTP: HTML title	5.9M
HTTP: Body Hash	50.8M
HTTP: Server	480K
HTTP: Header	22K
SSH: Host Key	14.3M
SSH: Banner	177K
VNC: Desktop Name	4.5K
SMTP: Banner	2.9M
FTP: Banner	1.5M
IMAP: Banner	144K
POP3: Banner	390K
CWMP: Header	10
CWMP: Body Hash	11
Telnet: Banner	219K
PPTP: Vendor	390K
MYSQL: Server Version	5.7K
Memcached: Server Version	129
MSSQL: Server Version	381
IPMI: Banner	116
IP's /16 subnetwork	37.3K
IP's ASN	67.7K

Table 1: GPS Features— GPS's features span all TCP protocols with an available banner on Censys (i.e., 15 unique protocols). GPS only uses network features that are most predictive of service presence; the filtering process is described in Appendix C. The dimensionality (i.e., number of unique values) is calculated using the Censys ground truth dataset described in Section 6.1.

5.2 Identifying Predictive Patterns

GPS's second phase is purely computational: building a probabilistic model to identify feature patterns most predictive of service presence. GPS uses the probabilistic model in the third and fourth stages of the process to scan for new services (Sections 5.3 and 5.4).

Feature selection. To identify feature values that are most predictive of service presence, GPS uses the seed set to extract the three categories of features—application, transport, and network—introduced in Section 4. GPS uses a total of 25 unique application and network layer features (Table 1) in our evaluation. GPS's features span all TCP protocols with an available banner on Censys (i.e., 15 unique protocols). GPS therefore accommodates the possibility that both popular (e.g., HTTP) and less popular (e.g., VNC) protocols contain information that are predictive of service presence. GPS's design allows for the user to easily include/remove any feature candidates.

Modeling interactions. GPS independently models different *interactions* of the three primary feature categories. This accommodates the possibility that any combination of the three feature categories may be predictive of Internet services. For example, knowing a host's network (Net_{IP}) can be less predictive when the host appears in many subnetworks (e.g., Android TVs) or more predictive when a service is in only one subnetwork (e.g., Freebox devices only appear in the Free network [4]). Concretely, GPS models the following:

1. Transport layer correspondence: the probability that a host will have $Port_a$ open given the host has $Port_b$ open (Expression 4).

$$\mathbb{P}(Port_a|Port_b) \tag{4}$$

2. Transport and application layer correspondence: the probability that a host will have $Port_a$ open given the host has $Port_b$ open and the service on $Port_b$ contains a specific application layer feature value (Expression 5).

$$\mathbb{P}(Port_a|(Port_b, App_{Port_b})) \tag{5}$$

3. Transport and network correspondence: the probability that a host will have $Port_a$ open given the host has $Port_b$ open and the host responding on $Port_b$ resides in network Net_{IP} (Expression 6).

$$\mathbb{P}(Port_a|(Port_b, Net_{IP})) \tag{6}$$

4. Transport, application and network correspondence: the probability that a host will have $Port_a$ open given the host has $Port_b$ open and the service on $Port_b$ contains a specific application layer feature value and the host responding on $Port_b$ resides in network Net_{IP} (Expression 7).

$$\mathbb{P}(Port_a|(Port_b, App_{Port_b}, Net_{IP})) \tag{7}$$

Note that all conditional probabilities rely on $Port_b$ having already been discovered, such that the relevant application layer and/or transport layer feature values can be used to predict $Port_a$. However, not all hosts respond on more than one port and not all hosts have a priori available information (e.g., a known service). We show in the following section how GPS uses its probabilistic models to predict services no matter how many ports the hosts responds on or how much a priori host information is available.

Computational scalability. We note that GPS's method for modeling feature interactions is computationally expensive; GPS must calculate all possible combinations of features that appear in the seed set. Furthermore, while the port feature ($Port_a$) has a dimensionality of 65K, application-layer features can vary widely in dimensionality (Table 1), thus allowing for potentially billions of feature-value combinations. Nevertheless, GPS's method is computationally scalable across all 65K ports because computing conditional probabilities is a parallelizable computation across all 65K ports. We discuss our implementation of the probabilistic model in Section 5.5 and how we use serverless computing to dramatically reduce the wall-clock time and use matrix operations to scale computations. We formally evaluate the computational complexity of GPS in Section 6.5.

5.3 Predicting The First Service

Predicting a host's first service must be treated differently than predicting the remaining services on the host. When predicting a host's first service, only network-layer features (i.e., the IP addresses' subnetwork) are available to predict service presence. In contrast, when at least one service is known, the information provided by that service can be used to predict other services on the same host (e.g., port correlations). Since the seed set only contains

information about a subset (e.g., 1%) of all hosts, GPS must first discover at least one service on each responsive IPv4 host in order to use it to further predict other services on the same host.

Scanning step size. As presented in Section 4, services are more likely to appear together in subnetworks. Thus, given a responsive service (IP, p) in the seed set, GPS must probe the IP addresses around IP (i.e., in its network) on port p, to maximize its chances of finding other services.

GPS faces a trade-off when building an efficient scanning strategy: how exhaustively should the subnetwork of a responsive service in the seed set be scanned? Scanning 100% of the IPv4 address space on a port increases the likelihood of finding all hosts that respond on that port, but requires more bandwidth and increases the impact on destination networks. Scanning, for example, the /24 subnetwork of IP IP on port p from the seed set requires substantially less bandwidth, but potentially misses hosts outside of the subnetwork that also respond on port p. The user's bandwidth constraint is the deciding factor of how large of a "scanning step size" (i.e., the subnetwork size to exhaustively scan a port) GPS should use, and is left as a user-specified parameter. We formally evaluate the bandwidth/coverage trade-off in Section 6.2 to help inform the user what scanning step size to use.

Relying on selective random probing of a particular network and port is a bandwidth-expensive but initially unavoidable process. Thus, GPS uses it only when absolutely necessary: predicting on each host *only* the service(s) that must be found first in order to predict any remaining services.

Choosing the most predictive services. GPS prioritizes finding the minimum set of services that is most informative for predicting any remaining services. For example, for a host that responds with a generic HTTP page on port 80 and a vendor-revealing banner on port 222, discovering the banner is likely much more predictive of the HTTP/80 service than vice versa (most HTTP/80 services do not respond on SSH/222 but most SSH/222 services do respond on HTTP/80). GPS uses its probablistic models (Equations 4–7) to calculate which service for each host in the seed set is most predictive of all the host's remaining services. If a host only responds on one port in the seed set, the sole service is the first and only service that must be predicted.

GPS executes the following algorithm to determine which subnetworks and ports to scan in order to prioritize finding the minimum set of most predictive services:

(1) For all hosts that respond on only one port in the seed set, save the service's $(Port_a, Net_{IP})$.
(2) For all hosts that respond on more than one port in the seed set, compute all four conditional probabilities (Equations 4–7). For every $(IP, Port_a)$ in the seed set, identify the $Port_b$ that results in the maximum $\mathbb{P}(Port_a)$, and save the $(Port_b, Net_{IP})$.
(3) Across all hosts, group together the required $(Port, Net_{IP})$ tuples and count the number of unique services they help predict in the seed scan (i.e., maximal coverage).
(4) Sort the $(Port, Net_{IP})$ based on maximum coverage.

Priors scan list. GPS's algorithm outputs a "priors scan list": an ordered list of unique tuples, consisting of a (port, subnetwork of size scanning step) pair (e.g., (80, 1.1.0.0/16)) in order of maximum coverage. The prior scans list allows GPS to collect the most predictive services across all ports in parallel. Note that not all ports and subnetworks will be scanned: only the (port, subnetwork) tuples that result in the maximum probability of all remaining services being found. Scanning the priors list allows GPS to find the most predictive service on each host, which it can next use to predict additional services on each host.

5.4 Predicting Additional Services

After GPS finds at least one service per host, GPS extracts three categories of features—application, transport, and network—in each discovered service to predict additional new services on the same host. GPS predicts new services by using the existing probabilistic models (Equations 4–7) to create a "most predictive features" list, and uses that list to predict additional services.

GPS uses the following algorithm to predict additional services:

(1) For each service in the seed set, identify the feature-tuple (e.g., $(Port_b, App_{Port_b})$) that results in the maximum $\mathbb{P}(Port_a)$. Save the feature tuple and predicted port, $Port_a$, in a "most predictive feature values" list. To account for services that appear on random ports and have a low maximum $\mathbb{P}(Port_a)$), discard all probabilities below 0.00001, which is roughly the hit rate of randomly probing the majority of ports.
(2) For each responsive service found in the *priors* scan, extract available feature values (e.g., TLS certificate, SSH host key).
(3) For each responsive service in the *priors* scan, and for all of its feature values, if the feature appears in the "most predictive feature values" list (i.e., $Port_b$, App_{Port_b}), save the predicted port $Port_a$ and the host's IP to a "predictions" list.

GPS's algorithm outputs a "predictions list": an ordered list of unique IP addresses and ports to scan. Step 1 is crucial to the GPS's algorithm; by using every service in the seed scan to build the "most predictive features" list, GPS's prediction algorithm guarantees that every service—that GPS has seen before in the seed set and is "predictable"—will be predicted by using the feature value pattern most likely to find the service.

5.5 Implementation

We describe an implementation of GPS that capitalizes on its parallelizable algorithm to drastically reduce the wall-clock time of predicting services.

To perform Internet-wide scans (e.g., collect a seed scan, scan for prior services, scan for predicted services), GPS chains together three existing tools to conduct a scan: ZMap [21] + LZR [25] + ZGrab [12]. ZMap is a stateless Layer 4 scanner (i.e., "SYN-scanner") that initiates TCP connections with Internet hosts in the GPS pipeline. GPS's use of ZMap allows it to easily be blocked by network operators, due to its unique fingerprint (IP ID = 54321). LZR then takes over the TCP connection, filters out middleboxes, and efficiently fingerprints services (a necessary step when scanning unassigned ports). For all services that are fingerprinted to be running real services, GPS provides LZR the option to forward the connection information to ZGrab, which can then complete the full Layer 7 handshake to collect additional application layer features.

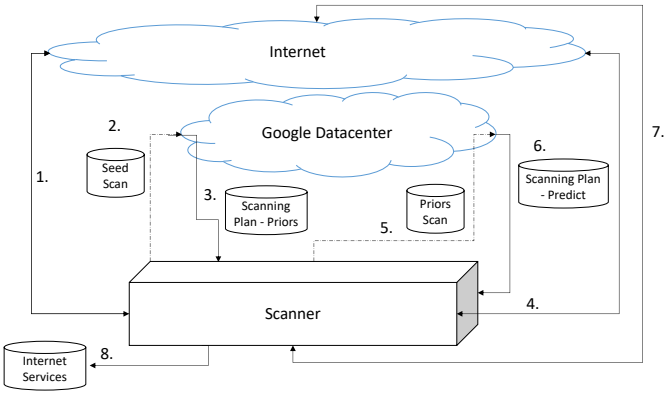


Figure 1: GPS Implementation—GPS's parallelizable prediction algorithm takes advantage of Google BigQuery's serverless platform to compute predictions in a highly parallelizable execution environment.

Implementing GPS's algorithm for identifying predictive patterns (Section 5.2) presents a challenge: finding all pairwise combinations of features and ports—although parallelizable—is computationally and memory intensive. Thus, to minimize wall-clock time, GPS can directly benefit from a highly parallelizable execution environment that, ideally, would be available to any user of the system. While GPS can be deployed on any server infrastructure, we find that Google BigQuery, a serverless database service that enables scalable analysis of petabytes of data [1], is well suited for the task. Implementing GPS's algorithms (e.g., calculating conditional probabilities, identifying the most predictive features) in a database query language is a natural choice, as the algorithms rely heavily on reading data, aggregating, and joining among shared data fields. We formally evaluate the implementation in Section 6.5 and show how with and without a highly parallelizable environment, GPS is 5870× and 5× faster than prior work, respectively.

We illustrate GPS's interaction with BigQuery in Figure 1; GPS uses BigQuery as a computational engine and does not rely on any pre-existing BigQuery data. Upon the completion of the seed scan, GPS uploads the results of the scan to BigQuery. Service features are extracted in BigQuery by either (1) selecting the appropriate fields from the uploaded scans, (2) performing operations on the IP address to extract the host's subnetwork, or (3) joining on a database that provides the feature (e.g., ASN). To efficiently calculate conditional probabilities, GPS uses BigQuery's SQL language to compute the pairwise co-occurence matrix for every feature and port, which involves JOIN-ing the dataset on itself to find all pairwise combinations of features and aggregating together identical feature value patterns and target-ports to calculate the conditional probabilities. GPS also uses BigQuery's SQL language to implement the strategy for predicting the first service (Section 5.3), by joining the seed scan with a computed co-occurence matrix.

Once the strategy for predicting the first service is calculated, GPS downloads the strategy on its scanning host. The scanning host exhaustively scans the (port, subnetwork) tuples and uploads the results of the scan to BigQuery, which again extracts the host features. GPS uses BigQuery's SQL language to create the most predictive feature list (Section 5.4) by joining the priors scan with the computed co-occurence matrix, and downloading the prediction strategy (i.e., a list of IPs and ports) for all remaining services to the scanning host. The scanning host then scans all remaining predicted services, and saves the results in a local database. We have released GPS's implementation under the Apache 2.0 License at https://github.com/stanford-esrg/gps.

6 EVALUATION

In this section, we evaluate GPS's ability to find services across all ports. First, we evaluate GPS's ability to find the maximum number of services under a variable bandwidth constraint. We show that, as services become harder to predict, the bandwidth required to find services increases: GPS finds 92% of services across all ports (with greater than two responsive IP addresses) with 131× less bandwidth compared to exhaustive scanning, but saves only 10× the bandwidth when finding 96% of services across all ports. Second, we evaluate GPS's precision. When minimizing Internet scanning's impact on available Internet resources, we show GPS is two orders of magnitude more precise than exhaustive probing. Third, when scanning popular ports, we compare GPS's accuracy and bandwidth with the XGBoost scanner [36]. GPS saves up to 28 times, and on average 2.3 times, the bandwidth required to achieve the same coverage of services. Fourth, we evaluate the computational complexity of predicting services and show that, compared to XGBoost scanner, GPS computes predictions 5 times faster when using a single core and up to *four orders of magnitude* faster when using BigQuery. Lastly, using GPS's predictions, we identify the feature values that are most predictive of service presence.

6.1 Methodology

Properly evaluating GPS requires finding an appropriate ground truth dataset and appropriately configuring GPS.

Approximating ground truth. Finding a "ground truth" dataset to evaluate GPS presents a challenge: no method exists to efficiently scan all 65K ports at 100% (hence the need for GPS). Thus, we evaluate GPS using two data sets. First, we use Censys' universal data set comprising of 100% IPv4 scans of the most popular 2K ports [11] sampled on July 26, 2021, which is the largest publicly available dataset that scans the most number of ports at 100%. Censys shares that, given their scanning bandwidth, it would require 196 days to exhaustively scan all 65K ports.

Censys targets only the most popular ports. Thus, we additionally evaluate GPS against a 1% random scan of the IPv4 address space across all 65K ports in April, 2021, requiring all 30 days to collect, using the LZR scanner [25]. While the LZR dataset spans all ports, it reduces the available sample for each port, which may miss patterns exhibited by a small number of hosts. We filter both dataset for real services using the steps described in Appendix B.

To create seed-scans and test sets for each dataset, we randomly assign each IP address, and its accompanying services, to either a seed or test set. Thus, a 2% Censys seed set leaves a 98% test set, and a 0.5% LZR seed set leaves a 0.5% test set. Although we spend an entire month scanning 1% of the IPv4 address space across 65K ports, responsive services are still sparse across most ports. Since we do not expect GPS to learn and predict services on ports with no

training data, we filter both the LZR seed and test set to only include ports that have greater than two responsive IP addresses. When using a 0.5% seed set, filtering leaves a remaining 13,162 ports.

Parameter tuning. GPS requires specifying a seed and step size as input parameters. We evaluate how the seed and scanning step size impact GPS's performance and find that a small step size increases GPS's precision, but decreases recall (Appendix D). This trade off happens because, as discussed in Section 5.3, scanning a smaller subnetwork of the prior port requires substantially less bandwidth, but potentially misses hosts outside of the subnetwork that could hold informative feature values. Increasing the seed size increases the fraction of *normalized* services found, since a larger seed size is more likely to encounter uncommon patterns that dominate uncommon ports. However, seed size does not substantially affect the fraction of overall services found. We specify in each experiment the seed and scanning step sizes chosen.

Features. We specify GPS to use 25 features (Table 1).

6.2 GPS's Coverage Across All Ports

GPS's objective is to maximize the number of services found across *all* ports. To quantify coverage, we use two metrics: fraction of all services found (Equation 1 in Section 3) and fraction of normalized services found (Equation 2 in Section 3), which weighs services on popular and unpopular ports equally. We calculate both metrics across both the Censys (2% seed set) and LZR (0.5% seed set) datasets. As a reference point, we also plot (1) the bandwidth used by an oracle predictor that knows exactly which services to probe (i.e., 100% accuracy) and (2) the fraction of services found as a function of bandwidth when exhaustively probing the minimum subset of ports that maximizes service discovery (i.e., port 80, (80,443), (80,443,7547), etc) . In comparison to exhaustively scanning all ports, this "optimal port-order" probing benchmark provides a tighter estimate for the minimum number of ports that must be exhaustively probed in order to find a maximum fraction of services. For example, exhaustively scanning only the 10 most popular ports (i.e., 0.015% of all ports) is the minimum set of ports that finds 5% of all services. We report bandwidth usage in units of 100% scans (i.e., 3.7 billion packets), in order to easily compare GPS performance with exhaustive scanning.

The number of services GPS finds depends on the bandwidth budget (Figure 2). When evaluating against 100% IPv4 scans across 2K ports (Censys), GPS initially finds 94% of services using 21× less the amount of bandwidth compared to optimal port-order probing. However, since GPS scans services in descending order of predictability, predicting services quickly becomes more bandwidth consuming: finding 98.2% of services across 2K ports only saves 7.6× the bandwidth compared to optimal port-order probing. Censys shares that they probe the equivalent of 572 100% scans every day to curate their dataset; GPS uses 3×10^{11} probes, or 6.3× less probes than Censys, to finds 98.2% of services across 2K ports.

The bandwidth coverage trade off is exacerbated when finding normalized services: GPS finds 46% of normalized services across 2K ports using 100× less bandwidth than optimal port-order probing but, when finding 67% of normalized services, saves only 50% of bandwidth. When evaluating against 1% IPv4 scans across all ports (with greater than 2 responsive IP addresses), GPS finds 92.5% and 95% of all services using 6× less and 2× less bandwidth, respectively, than optimal port-order probing. GPS's performance drop when evaluating against all ports is likely due to the 4x smaller seed size; GPS's performance is nearly identical when using a 0.5% training seed set across 2K ports (Appendix D.2).

We quantify the fundamental limitations that GPS faces when predicting services in Section 7.

6.3 GPS's Precision Across All Ports

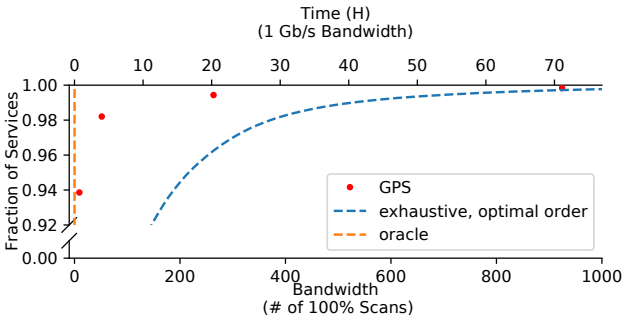
When minimizing Internet-wide scanning's impact on destination networks, we show that GPS is over two orders of magnitude more precise than random probing. GPS scans services that are most predictable first (Section 5.4). Thus, GPS requires less bandwidth, is more precise, and minimizes time-to-discovery to find the most predictable services at the beginning of the scanning schedule. We use the Censys dataset, to evaluate against 100% scans, with a 1% seed size. To maximize GPS's precision, we configure GPS with a small (/20) scanning step size. We plot in Figure 3 GPS's precision as it continues to find services, and, for comparison, show the precision of exhaustively probing all ports in the optimal order that prioritizes discovering the most number of services first.

GPS is able to find the first 1% of all services with a 36% precision—one order of magnitude more precise than exhaustive probing. GPS finds up to 94% of all services and 46% of normalized services while being consistently over an order of magnitude more precise than exhaustive probing. For example, GPS finds the 94th-percentile of services with 204× more precision than exhaustive probing. Note that GPS's precision decreases over time as it continues to exhaust its predictions in descending order of predictability. Once GPS exhausts all predictions, it can be optionally configured to randomly probe the rest of all previously un-probed services, thereby eventually finding 100% of all normalized services (albeit at a very slow rate).

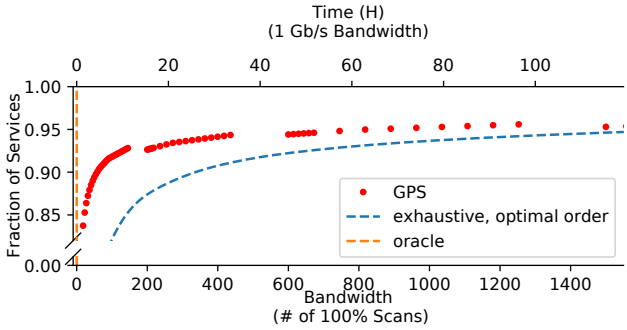
6.4 GPS vs. Machine Learning

As introduced in Section 2, Sarabi et al. [36] use a state-of-the-art XGBoost classifier [18] to predict responsive IPs for a given port. Unfortunately, the classifier cannot be deployed across all 65K ports due to (1) its prohibitively expensive runtime and (2) the lack of 10 million IP address samples across 99.99% of all 65K ports to train robust models [36]. Nevertheless, we benchmark GPS against the XGBoost since it is the closest related work.

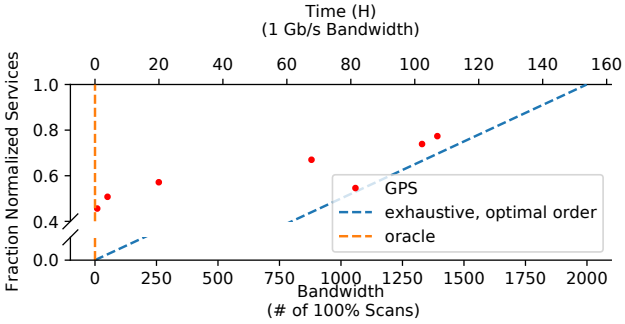
Methodology. Since Sarabi et al.'s XGBoost scanner is not open sourced, we benchmark GPS using the results presented in their paper. We use XGBoost scanner's best performing implementation: a sequence of models that follow an optimal ordering of port scanning and use port responses as input features. We configure GPS to use a seed size of 0.5% (equivalent to the number of IPs Sarabi et al. use to train their model) from the Censys dataset—also used by Sarabi et al. —for evaluation. We configure GPS to use a /16 step size in order to balance coverage and accuracy. We provide an evaluation for all 19 TCP ports and protocols evaluated by Sarabi et al. Since their work provides metrics across varying step-sizes of coverage,



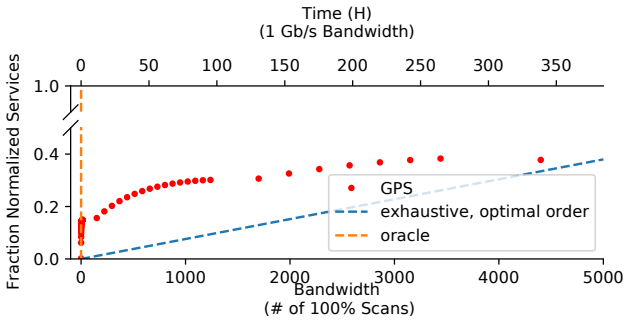
(a) Service Discovery (Censys): GPS finds 94% of services using 21× less the amount of bandwidth compared to optimal port-order probing (2K ports, 100% scan, 2% seed).



(b) Service Discovery (LZR): GPS finds 92.5% and 95% of all services using 6× less and 2× less bandwidth, respectively, than optimal port-order probing. (all ports, 1% scan, 0.5% seed).



(c) Normalized Service Discovery (Censys): GPS finds 46% of normalized services across 2K ports using 100× less bandwidth than optimal port-order probing, but when finding 67% of normalized services, only saves 50% of bandwidth. (2K ports, 100% scan, 2% seed).



(d) Normalized Service Discovery (LZR): GPS finds 17% and 38% of normalized services using 15× and 1.7× less bandwidth, respectively, than optimal port-order probing. (all ports, 1% scan, 0.5% seed).

Figure 2: Finding Services—Under a variety of training and testing sets, GPS is able to find up to 85% of normalized services and 99.8% of all services using less bandwidth than optimal port-order probing.

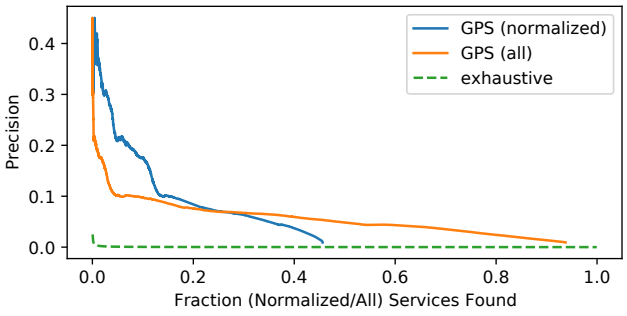


Figure 3: GPS Precision—GPS finds up to 94% of all services and 46% of normalized services while being consistently over an order of magnitude more precise than exhaustive probing.

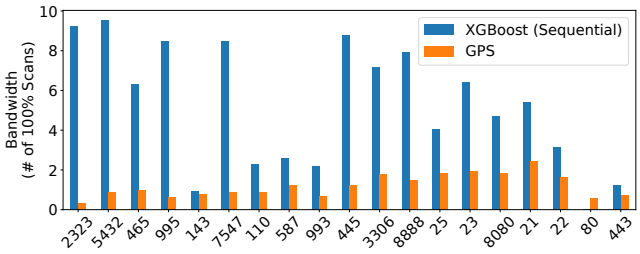
we evaluate XGBoost scanner at the maximum coverage level that GPS can achieve[1] (i.e., a 98.8% average coverage).

[1]Using GPS with a 0.5% seed size and /16 step size, GPS achieves the following maximum coverage: 99.9% for ports 21,22,80,443, 99% for ports 23, 25, 119, 445, 465, 587, 993, 3306, 7547, 8080, 8888; 98% for ports 143, 995, 5432; and 94% for port 2323.

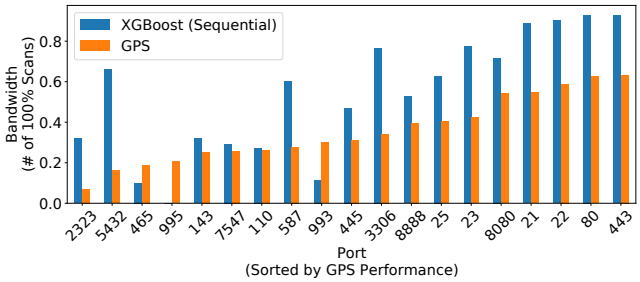
Similar to how GPS scans a minimum set of predictive services across all hosts to find remaining services, XGBoost scanner also relies on a set of prior scanned IP addresses and ports to predict additional services. In Figure 4a we plot the bandwidth required by the XGBoost scanner to collect all needed prior information (following their optimal scanning sequence) and compare it to the bandwidth required by GPS to scan the minimum set of predictive services. We note that, while Sarabi et al. do not directly reveal their model's prediction accuracy, bandwidth and accuracy are correlated: the less bandwidth needed to accurately predict services, the more accurate the model's predictions are.

Across all ports, GPS requires an average 5.7x less bandwidth to collect its minimum set of predictive services and, at best, 28x less bandwidth when scanning port 2323. Scanning port 80 is the only case in which GPS requires more bandwidth than the XGBoost scanner; XGBoost scanner does not rely on a minimum set of predictive services to predict services on port 80 (i.e., it only uses network layer features to immediately predict all services).

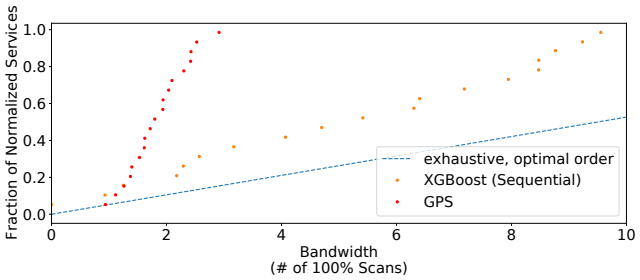
Assuming that the minimum set of predictive services has been scanned (e.g., a user originally intended to scan more than one port), we show in Figure 4b the bandwidth required by each system to



(a) Bandwidth Used to Scan Minimum Set of Predictive Services—GPS saves up to 28x more bandwidth than the XGBoost scanner.



(b) Bandwidth Used to Scan Remaining Services—GPS saves more bandwidth than XGBoost scanner when scanning 16 of 19 popular ports.



(c) Normalized Service Discovery—GPS uses 3x less bandwidth to find 98.5% of all normalized services than XGBoost scanner.

Figure 4: GPS vs. XGBoost Bandwidth Consumption

achieve majority coverage[1] of a target port. GPS requires less bandwidth than XGBoost when scanning 16 out of the 19 evaluated ports. For example, GPS requires 4x less bandwidth when predicting hosts on ports 2323 and 5432. Ports 995, 993 and 465 are the only ports that XGBoost requires less bandwidth to scan. However, XGBoost's performance gains are not free: the XGBoost scanner require 3–14x more prior bandwidth to collect the minimum set of predictive services for ports 995, 993 and 465 compared to GPS (Figure 4a). On average, GPS requires half the amount of bandwidth to achieve the same coverage of a target port. In other words, simple conditional probabilities, on average, achieve a greater accuracy than the XGBoost machine learning classifier when predicting services.

When accounting for all the bandwidth needed to find services, we show in Figure 4c how GPS finds 98.5% of all normalized services using 3 times less bandwidth than the XGBoost scanner.

6.5 Computational Complexity

Given the appropriate computational resources, GPS's parallelizable prediction algorithm allows it to find services across all ports in a constrained amount of wall time. In this section, we discuss how GPS's performance is dependent upon the availability of a seed scan, bandwidth, and computational resources. We report the breakdown of GPS's performance in Table 2 when configuring GPS to use all 25 features in Table 1, a 1% seed scan and a /16 scan step size to predict services across all ports.

Time. GPS spends time on three categories of tasks: scanning, prediction, and data transfer. GPS's bottleneck lies in bandwidth. Without an existing seed scan, GPS requires 12.3 days to perform all scans, which is bounded by existing scanning tools and available bandwidth. Collecting the initial 1% seed scan contributes to 97.5% of all scanning time, due to the low precision when randomly probing (i.e., below 0.00001 for the majority of ports). However, if a seed scan is already available, GPS can forego collecting the initial seed scan, reducing the overall runtime by 94%. In contrast to the seed scan, both prediction scans take only hours to execute, due to their orders of magnitude higher precision than exhaustively probing uncommon ports. In total, GPS predicts 28 billion services that require 8 hours to scan.

On a single CPU core, GPS performs predictions in roughly 9 days and 9 hours—5.6x faster than the XGBoost scanner on a single GPU. However, GPS's wall time can be drastically reduced when marrying its parallelizable prediction algorithm with a highly parallelizable computing environment. When implementing GPS's prediction algorithm to use BigQuery, GPS computes all predictions within 13 minutes of wall-time, *4 orders of magnitude* less time than the XGBoost scanner, while costing only 75 cents.

Unfortunately, computing over big data on any platform often faces a data transfer bottleneck. Using a serverless platform requires that data be uploaded and downloaded to/from the platform. GPS needs a total of 9.1 hours to up/download a total of 606 GB of data to BigQuery. The biggest bottleneck in the up/download process is the up/download bandwidth, which GPS experiences to be between 18MB/s–30MB/s when using 24 parallel processes. BigQuery does not charge for inbound data transport, leaving the total BigQuery cost to be bounded by computation. Sarabi et al. do not report the time it takes to load all necessary data into XGBoost scanner's GPU, which is often considered to be the biggest bottleneck for large-scale machine learning computations [6].

Space. The amount of memory GPS requires is directly correlated with the size of the seed scan, the number of features used to predict services, and the implementation used to calculate conditional probabilities. The seed scan itself—including all of its features—is often quite small; a filtered[2] 1% IPv4 65K port seed scan collected by LZR [25] is only 4 GB. However, GPS requires substantially more memory when predicting services, which is determined by the implementation of conditional probability calculations. For example, the implementation described in Section 5.5— JOIN-ing the data on itself to find all pair-wise combinations of an IP's features—increases the memory footprint by at least 50 fold relative to the

[2]Using the methodology in Appendix B.

	Bandwidth	Computation Time (Single Core)	Wall-clock Time	Data Processed/Shuffled	Cost
1% Seed Scan (if needed)	1.5 Gb/s	–	12 Days	–	–
Seed Scan Upload	20 Mb/s	–	3.5 Min	4 GB	0¢
Predicting First Service (PFS)	-	6 Days 2 Hours	8 Min (BigQuery)	4 TB	13¢
PFS Download	1.3 KB/s	–	7 Sec	9.3 KB	0¢
PFS Scan	50 Mb/s	–	20 Min	–	–
PFS Scan Upload	30 Mb/s	–	34 Min	55 GB	0¢
Predicting Remaining Service (PRS)	–	3 Days 7 Hours	5 Min (BigQuery)	2.5 TB	62¢
PRS Download	18 Mb/s	–	8.5 Hours	547 GB	0¢
PRS Scan	50 Mb/s	–	8 Hours	–	–
Total Scanning Wall-Time	–	–	12.3 Days	–	–
Total Download/Upload Wall-Time	–	–	9.1 Hours	–	–
Total Computational Wall-Time	–	–	13 Min	–	–
Total	–	9 Days 9 Hours	12.7 Days	7 TB	75¢

Table 2: GPS Performance Breakdown—When using a 1% seed scan and a /16 scan step size to predict services across all ports, GPS bottleneck lies in bandwidth. GPS computes all the necessary predictions within 13 minutes when using BigQuery. Due to GPS's high scanning precision, both prediction scans use a substantially lower scanning rate (i.e., 50 Mb/s compared to 1.5Gb/s) in order to avoid packet congestion and incoming packet drop.

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%

Table 3: Top 5 Predictive Features—A port's protocol is the most predictive feature, predicting 18.7% of normalized services.

seed scan and the number of features used. GPS also requires sizeable disk space when outputting the final list of predicted services; writing the predicted 28 billion IP and port pairs results in 547 GB of output.

Researchers deploying GPS will achieve the largest performance gains when using a highly parallelizable computational environment with two orders of magnitude more memory than the initial seed scan size.

6.6 Which features are most predictive?

By using conditional probabilities, it is simple to understand what features are most predictive of Internet services. When running GPS using a 1% seed set to predict services across all ports in the Censys dataset, GPS selects 402K unique feature *values* as being most predictive of a service. Information found when using the HTTP protocol (e.g., HTTP header, HTTP Body hash) is most predictive compared to every other protocol, contributing to 45% of the most predictive features values. We present the top-5 most predictive feature candidates that GPS identifies in Table 3. Across 18.7% of normalized services, the protocol (e.g., SSH) running on a host's port (i.e., (Port, $\text{Port}_{\text{Protocol}}$)) is most predictive of another port being responsive.

GPS identifies 64 unique tuples of feature *values* that are most predictive of services, which include the interaction of application-layer and network-layer features such as: (ASN, TLS certificate) (4.4% normalized services), (ASN, SSH Key) (1.2%), VNC name (0.4%), and (ASN, FTP banner) (0.24%). For example, 95% of hosts in Distributel Network (ASN 1181) that respond on port 23 with the telnet banner "Telnet service is disabled or Your telnet session has expired due to inactivity..." host HTTP content on port 8082; and 98% of hosts in Bizland (ASN 29873), which respond with an IMAP banner requesting TLS on port 143, also host SSH on port 2222. These findings show that GPS's ability to model the interaction of features (Section 5.2), and decision to not exclude any feature candidates, helps predict services on various networks and ports.

7 LIMITATIONS

While GPS dramatically shifts the barrier for scanning all Internet services, there remain existing challenges.

Pattern Mining. GPS is bounded by the features it is configured to use (Appendix C) and the resulting patterns it mines. If, for example, detecting a pattern relies on collecting features from the union of multiple port responses, GPS may not detect the pattern due to the computationally expensive nature of calculating correlations between more than two ports. Nonetheless, with the advent of increased availability of computational parallelism, GPS's algorithm is modular and easily extensible to add additional feature correlation computations.

IPv6. GPS relies on exhaustive scans to obtain a set of responsive IP addresses to use for service predictions: this approach does not work for IPv6 due to the larger search space. However, given known IPv6 addresses that respond on at least one port, GPS can be used to predict other responsive services on the known IPv6 addresses.

Random Host Configuration. While services often exhibit predictable patterns, random configurations of hosts will always present

a limitation for predicting services. For example, the manual of the most common (21%) IoT device found in our LZR scan, FRITZ!Box, states that "for security reasons, FRITZ!Box sets up a random TCP port for HTTPS when internet access via HTTPS is enabled" [2, 5]. Furthermore, routers can easily port-forward services through random ports [7, 10]. We find in our LZR scan that at least 55% of services are likely being port-forwarded (i.e., different TTL values returned across all services being hosted) across 99% of the most uncommon ports.

To understand how random host configurations impact GPS, we set up an experiment in which we: (1) Use a 95% seed set to predict the remaining 5% of services of the Censys dataset, thereby assuming that nearly all patterns are known beforehand; (2) Count all services on an IP as being discovered the moment at least one service has been discovered on an IP, thereby assuming that feature correlations are 100% available and 100% accurate; and (3) Specify the largest scanning step size (/0) to maximize the total fraction of normalized services found. Under these ideal conditions, 80% of all normalized services can be discovered using less bandwidth than exhaustive scanning, a percentage slightly lower than GPS due to the small size of the random test set. These results illustrate how, in a "real-world" setting, GPS performs near what is best achievable and illustrates what fundamental limitations lie ahead for any intelligent Internet-wide scanning system.

8 CONCLUSION

In this work, we introduced GPS, an intelligent scanning system that scalably and efficiently predicts services across all IPs and ports with no prior knowledge. We demonstrated how a seemingly simple predictive framework based conditional probabilities can perform orders of magnitude faster and with more accuracy than the leading machine learning implementation. GPS finds 92.5% of services across all ports using 131× less bandwidth than exhaustive scanning, while being 204× more precise. By releasing GPS as an open source tool, we hope that the research community will now be able to find the billions of previously-missed services, at a fraction of the cost of exhaustive scanning.

ACKNOWLEDGEMENTS

The authors thank Tatyana Izhikevich, Katherine Izhikevich, Kimberly Ruth, Deepak Kumar, Pratiksha Thaker, Deepti Raghavan, members of the Stanford University security and networking groups, our shepherd, Lixia Zhang, and the anonymous reviewers for providing insightful discussion and comments. This work was supported in part by the National Science Foundation under award CNS-1823192, Google., Inc., the NSF Graduate Fellowship DGE-1656518 and a Stanford Graduate Fellowship.

REFERENCES

[1] Bigquery: Cloud data warehouse. https://cloud.google.com/bigquery. Accessed: 2021-08-28.
[2] Cannot access user interface via HTTPS from the home network. https://en.avm.de/service/knowledge-base/dok/FRITZ-Box-4020/3439_Cannot-access-user-interface-via-HTTPS-from-the-home-network/. Accessed: 2022-01-26.
[3] Dataset for LZR: Identifying unexpected Internet services. https://scans.io/study/lzr. Accessed: 2022-01-26.
[4] Free. https://www.free.fr/freebox/. Accessed: 2021-09-09.
[5] FRITZ!box 7490 help. https://service.avm.de/help/en/FRITZ-Box-Fon-WLAN-7490/015/hilfe_internet_remote_https. Accessed: 2021-08-13.
[6] GPUs are fast! datasets are your bottleneck. https://towardsdatascience.com/gpus-are-fast-datasets-are-your-bottleneck-e5ac9bf2ad27. Accessed: 2021-09-06.
[7] How to port forward – general guide to multiple router brands. https://www.noip.com/support/knowledgebase/general-port-forwarding-guide/. Accessed: 2021-09-09.
[8] Moving beyond the noise by filtering Internet pseudo services. https://censys.io/blog/beyond-noise-by-filtering-pseudo-services/. Accessed: 2021-08-19.
[9] Recommenders. https://github.com/microsoft/recommenders. Accessed: 2021-09-06.
[10] Understanding port forwarding. https://stevessmarthomeguide.com/understanding-port-forwarding/. Accessed: 2022-01-26.
[11] Universal Internet BigQuery dataset. https://support.censys.io/hc/en-us/articles/360056063151-Universal-Internet-BigQuery-Dataset. Accessed: 2021-09-09.
[12] ZGrab 2.0 - GitHub. https://github.com/zmap/zgrab2. Accessed: 2019-12-14.
[13] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, et al. Understanding the Mirai botnet. In *USENIX Security Symposium*, 2017.
[14] S. Bano, P. Richter, M. Javed, S. Sundaresan, Z. Durumeric, S. J. Murdoch, R. Mortier, and V. Paxson. Scanning the Internet for liveness. *ACM SIGCOMM Computer Communication Review*, 2018.
[15] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P.-Y. Strub, and J. K. Zinzindohoue. A messy state of the union: Taming the composite state machines of TLS. In *IEEE Symposium on Security and Privacy*, 2015.
[16] R. Beverly. Yarrp'ing the Internet: Randomized high-speed active topology discovery. In *ACM Internet Measurement Conference*, 2016.
[17] S. Checkoway, R. Niederhagen, A. Everspaugh, M. Green, T. Lange, T. Ristenpart, D. J. Bernstein, J. Maskiewicz, H. Shacham, and M. Fredrikson. On the practical exploitability of dual EC in TLS implementations. In *USENIX Security Symposium*, 2014.
[18] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.
[19] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman. A search engine backed by Internet-wide scanning. In *ACM Conference on Computer and Communications Security*, 2015.
[20] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, et al. The matter of heartbleed. In *ACM Internet Measurement Conference*, 2014.
[21] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *USENIX Security Symposium*, 2013.
[22] P. Foremski, D. Plonka, and A. Berger. Entropy/IP: Uncovering structure in IPv6 addresses. In *ACM Internet Measurement Conference*, 2016.
[23] 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.
[24] C. A. Gomez-Uribe and N. Hunt. The Netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 2015.
[25] L. Izhikevich, R. Teixeira, and Z. Durumeric. LZR: Identifying unexpected Internet services. In *USENIX Security Symposium*, 2021.
[26] M. Kula. Metadata embeddings for user and item cold-start recommendations. *arXiv preprint arXiv:1507.08439*, 2015.
[27] D. Kumar, K. Shen, B. Case, D. Garg, G. Alperovich, D. Kuznetsov, R. Gupta, and Z. Durumeric. All things considered: An analysis of IoT devices on home networks. In *USENIX Security Symposium*, 2019.
[28] F. Li, Z. Durumeric, J. Czyz, M. Karami, M. Bailey, D. McCoy, S. Savage, and V. Paxson. You've got vulnerability: Exploring effective vulnerability notifications. In *USENIX Security Symposium*, 2016.
[29] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker. Unexpected means of protocol inference. In *ACM Internet Measurement Conference*, 2006.
[30] Y. Ma, B. Narayanaswamy, H. Lin, and H. Ding. Temporal-contextual recommendation in real-time. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.
[31] B. Marczak, J. Scott-Railton, K. Berdan, B. Razzak, and R. Deibert. Hooking candiru: Another mercenary spyware vendor comes into focus. Technical report, 2021.
[32] B. Marczak, J. Scott-Railton, S. Prakash Rao, S. Anstis, and R. Deibert. Running in circles: Uncovering the clients of cyberespionage firm circles. Technical report, 2020.
[33] J. McInerney, B. Lacker, S. Hansen, K. Higley, H. Bouchard, A. Gruson, and R. Mehrotra. Explore, exploit, and explain: personalizing explainable recommendations with bandits. In *ACM conference on recommender systems*, 2018.
[34] G. C. Moura, C. Ganán, Q. Lone, P. Poursaied, H. Asghari, and M. van Eeten. How dynamic is the isps address space? towards Internet-wide dhcp churn estimation. In *IFIP Networking Conference*, 2015.

[35] A. Murdock, F. Li, P. Bramsen, Z. Durumeric, and V. Paxson. Target generation for Internet-wide IPv6 scanning. In *ACM Internet Measurement Conference*, 2017.
[36] A. Sarabi, K. Jin, and M. Liu. *Smart Internet Probing: Scanning Using Adaptive Machine Learning*. 2021.
[37] SHODAN. The search engine for Internet-connected devices. https://www.shodan.io/.
[38] K. Vermeulen, J. P. Rohrer, R. Beverly, O. Fourmaux, and T. Friedman. Diamond-miner: Comprehensive discovery of the Internet's topology diamonds. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2020.

APPENDIX

Appendices are supporting material that has not been peer-reviewed.

A RECOMMENDATION SYSTEMS FOR INTELLIGENT SCANNING

Proprietary recommendation systems have successfully recommended millions of items to millions of users at Netflix [24], Spotify [33], and Amazon [30]. The model's success inspires us to use a recommendation system to recommend/predict responsive ports for IP addresses. We explore over 30 different open source recommendation models [9] and find that while existing recommendation systems allow features to be assigned to users (i.e., IPs) and/or items (i.e., ports), they do not directly allow for features to be assigned to their interaction (i.e., IP,Port tuples). Consequently, application layer features for every service cannot be easily represented. Further, the majority of models do not support adding new features, which may have been discovered as a result of a successful prediction, without the undesirable computational overhead of re-training the model.

To evaluate if open source models can successfully predict services, we extend a popular open-sourced hybrid recommender system, LightFM [26] to predict a responsive port, given an IP address. We provide the implementation of the model, including all of its parameters, on GitHub[3]. Given the framework's inability to assign features to specific services (i.e., (IP, Port) pairs), we are only able to encode features for IP addresses and ports. We experiment with assigning different network layer features (e.g., autonomous system, /16 subnetwork, /20 subnetwork), to every IP address and assigning a binary feature (designating whether the port number is IANA assigned) to every port. We train the model on the LZR dataset across all 65K ports (using an 0.8% IPv4 seed set) and have it generate 100 port predictions for every IP address in the test set (i.e., similar to generating 100 "100% prediction scans"). The model finds a maximum of 47% of all services—consistently performing worse compared to exhaustively probing ports in an order the prioritizes finding the most number of services—and 1.5% of normalized services. Due to the feature constraints that recommendation systems impose and this model's poor performance, we choose to stop pursuing adapting recommendation systems for predicting services across all ports.

B FILTERING FOR REAL SERVICES

Prior work has shown that a substantial number of IP addresses host "pseudo services" [8]. Pseudo services are often HTTP or HTTPS webpages that have successfully loaded, but display a message stating that no services exists on the webpage itself. We conduct a LZR [25] scan across all 65K ports on a random 1% subset of the IPv4 address space in March 2021 and find that across 96% of all 65K ports, the vast majority of services belong to a host that serve pseudo services on greater than 1,000 contiguous ports.

To ensure GPS does not learn to predict "pseudo services," GPS's seed set must filter all pseudo services. Over 80% of pseudo services are simply filtered by first, removing expected dynamic fields (e.g., HTTP date field, HTTP Cookie field, TLS random bytes) in the data and secondly, removing all services on the host that share the same filtered data. However, the long tail of pseudo services are much harder to fingerprint and filter. For example, data containing an unexpected random string (e.g., "Incident ID", timestamp), results in the data and content-length to slightly differ. Existing filtering solutions are often complex and incomplete [29]. Thus, we filter any host that serves more than 10 services—a method identifies pseudo services with 100% recall and 99% precision. All results in this work assume that pseudo services are filtered.

[3]https://github.com/stanford-esrg/recommender-system-gps

Network Feature	% Services Most Predictive
ASN	36%
/16	20%
/18	8%
/19	8%
/17	8%
/20	7%
/21	6%
/22	4%
/23	3%

Table 4: Network Features—When configuring GPS to use the /16–/23 subnetworks of an IP address, the ASN and /16 of an IP address are most predictive for the majority of services.

C NETWORK-LAYER FEATURE CANDIDATES

The final implementation of GPS is configured to use only the network-layer features that are most predictive of the majority of services. Reducing the number of network-layer features reduces the computational complexity of GPS's predictive algorithm. To determine which network-layer features are most predictive of service presence, we initially configure GPS to use all subnetwork sizes between /23–/16, and the autonomous system number. GPS builds a probabilistic model using a 0.5% seed set from the LZR dataset, which we use to analyze which network features are most predictive across all services. We show in Table 4 how the IP's ASN and /16 subnetwork are most predictive of service presence. It is no surprise that larger subnetworks are more predictive of service presence: larger subnetworks are more likely to be shared amongst multiple hosts in the seed set.

D PARAMETER TUNING

D.1 Varying Step Size

GPS minimizes bandwidth during prediction by regulating the scanning step size. As the step size decreases, the precision of finding

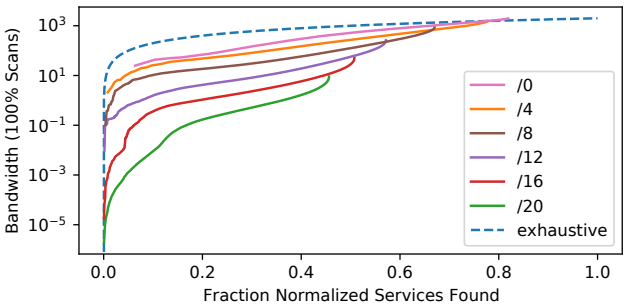
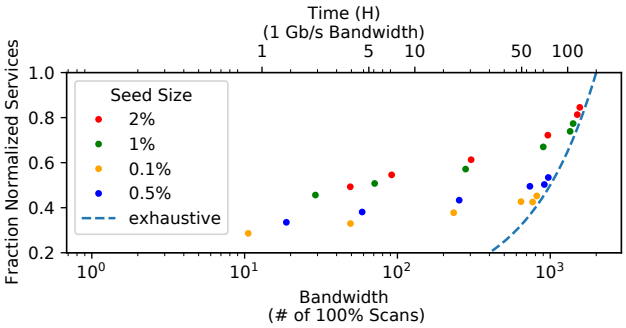


Figure 5: Varying Step Size (Censys)—A smaller scanning step saves more bandwidth when initially finding services, but ultimately finds less services compared to a larger scanning step size.

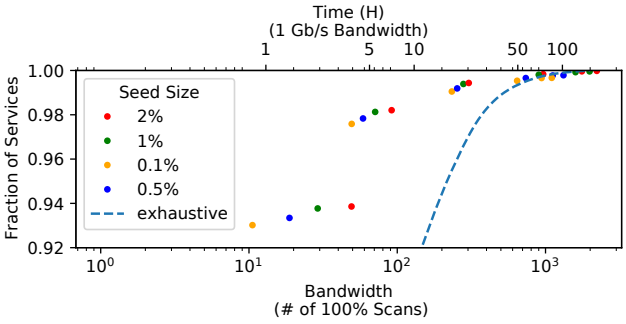
services increases, causing the overall bandwidth required to decrease. For example, as seen in Figure 5, finding the first 25% of normalized services using a scanning step size of /12 uses one order of magnitude more bandwidth compared to a scanning step size of /20. No GPS configuration finds more than 82% of normalized services with a bandwidth usage better than exhaustive probing.

D.2 Varying Seed Size

The seed size regulates the amount of unique service patterns GPS has already seen. We plot the amount of bandwidth needed (including collecting the seed size) and fraction of services found in Figure 6. When optimizing to find normalized services, Figure 6a shows that, for a bandwidth budget above 30 100% IPv4 scans, a 2% IPv4 seed size always finds the most fraction of normalized services compared to smaller seed size. This indicates that patterns found in a larger seed scan are crucial for finding normalized services. However, when optimizing to find the largest fraction of services, smaller seed sizes are sufficient (Figure 2a), indicating that the most predictive patterns for finding popular services can be detected with a small seed size.



(a) Normalized Service Discovery



(b) Service Discovery

Figure 6: Varying Seed Size (Censys)—A larger seed size increases the fraction of normalized services GPS finds, but does not substantially impact the fraction of all services GPS finds.