Bootstrapping of Peer-to-Peer Networks

Chris GauthierDickey
Department of Computer Science
University of Denver
chrisg@cs.du.edu

Christian Grothoff
Department of Computer Science
University of Denver
grothoff@cs.du.edu

Abstract

In this paper, we present the first heuristic for fully distributed bootstrapping of peer-to-peer networks. Our heuristic generates a stream of promising IP addresses to be probed as entry points. This stream is generated using statistical profiles using the IP ranges of start-of-authorities (SOAs) in the domain name system (DNS). We present experimental results demonstrating that with this approach it is efficient and practical to bootstrap Gnutella-sized peer-to-peer networks – without the need for centralized services or the public exposure of end-user's private IP addresses.

1 Introduction

The primary promise of peer-to-peer technology is the decentralization of services and various associated benefits. While peer-to-peer networks do not necessarily decentralize all functions for reasons of performance, simplicity, control and in particular security, it is generally desirable to have available efficient and effective means for fully decentralizing any common peer-to-peer operation.

One key operation in any open peer-to-peer overlay network is *bootstrapping*, the initial discovery of other systems participating in the network. Nascent peers need to perform such an operation in order to join the network. Bootstrapping does not include the maintenance of connections or exchange of topology information for peers that are already connected to the network at large. However, in our definition, bootstrapping does include operations needed to repair overlays that have split into disconnected subgraphs. To the best of our knowledge, no effective and efficient protocol for fully decentralized bootstrapping of open peer-to-peer networks has been proposed previously.

One obvious approach for fully decentralized bootstrapping is the method of brute-force scanning of the entire address space for existing peers. Scanning 4 billion IP addresses is clearly an expensive proposition; however, it has the advantage of being a completely decentralized operation. Given an estimated size of about 1.3 million peers for the Gnutella network [4], the chance of finding a peer is less than 0.03%. In our experiments, a brute-force random global scan for Gnutella peers requires on average 2425 attempts before finding the first peer. The simple trick of excluding unallocated and reserved IP ranges can almost double the chance of success.

In this paper, we will propose methods based on classification of IP address ranges using DNS that can help improve the success rates of this completely decentralized approach for peer-to-peer bootstrapping. The fundamental assumption of our research is that addresses in peer-to-peer networks have a signficant bias in their distribution across different organizations, as evidenced in Gnutella and Skype measurements [2, 3]. By biasing the scan towards organizations with a disproportionally high number of participants, we obtain an efficient and fully decentralized peer-to-peer bootstrapping method that is competitive when compared to approaches using somewhat outdated hostlists.

An extended version of this paper with a broader discussion of related work is available as a technical report at `http://grothoff.org/christian/pbtr.pdf`.

2 Approach

Our approach to peer-to-peer bootstrapping consists of two parts. First, a profile of the IP addresses of peers participating in the P2P network is generated. Using this list, a statistical profile is generated that describes, for each organization (as identified by DNS), the probability of how likely it is to find peers in the IP space of the organization. Second, the resulting statistical profile is used by peers to generate a stream of promising IP addresses for bootstrapping.

The specific method for obtaining a list of IP addresses of peers in the P2P network is dependent on the specifics of the network. In our experiments, we use a full graph traversal (Gnutella) [4], random walks (DirectConnect) and connection statistics from super-peers (E2DK). The resulting IP lists are matched against the start of authority (SOA)

for the respective IP address in DNS. In other words, the range of IP addresses of an organizations is identified as the range of IP addresses for which the same SOA is specified in the global DNS database. The P2P vendor then ships the resulting small database containing success probabilities for various organizations with the P2P software.

Using this statistical profile, the proposed approach provides peers that are trying to bootstrap with a randomized algorithm producing an infinite sequence of promising IP addresses that the peer should probe. The algorithm to generate IP addresses to scan works as follows. First, the algorithm uses a random number generator to generate three 8-bit values a, b and c, which are the the first 24 bits of an IP address of the form $a.b.c.x$. It then determines the probability of finding peers for the 256 possible values for $x \in [0:255]$ based on the SOA for the particular subnet. (While it is technically possible that the subnet is shared by multiple SOAs, this is hardly ever the case in practice.) Given a probability p of finding a peer in the address range of the the entire organization, the peer then selects at most $k = \lfloor p \cdot n \rfloor$ IP addresses in the subnet.[1] The parameter n is a trade-off between finding peers with few attempts and probing a diverse set of networks. In our implementation, the k values for x are determined using the equivalence class $x \equiv b \ mod \ p$ with $p = \lceil \frac{k}{256} \rceil$ with a randomly selected value for b. The smallest values for x are probed first – most organizations allocate IP addresses sequentially, making small values for x a bit more likely to result in active IP addresses.

There are various reasons why the proposed approach uses the SOA in order to determine the organization to which an IP address belongs to. First, the DNS names of large organizations are unlikely to change even as new IP addresses are allocated to an organization. Also, if the SOA of an IP address changed, it is likely that the corresponding IP address space was allocated to a different organization; naturally, the specific name of the SOA server may change without significant changes in the organization; only the domain name should be considered significant. Given that organizations are unlikely to run a DNS server for only a few IP addresses, using the SOA allows the client to determine the organization for an entire range of IP addresses with just a couple of DNS queries. The number of DNS queries is important since performing billions of DNS lookups would be worse than scanning billions of IP addresses. Finally, unlike hostnames, there is only one SOA for any given IP address.

3 Experimental Results

Experiments testing the proposed heuristic were performed between October 2007 and February 2008 using IP lists for Gnutella [1], E2DK (eMule) and DirectConnect. The Gnutella IP list was extracted from a topology crawl performed between September 2004 and August 2007 by Cruiser [4]. The IP addresses for E2DK and DirectConnect were taken from topology crawlers in October 2007. The number of IP addresses and their source are listed in Table 1.

[1] If $p \geq \frac{256}{n}$, all values for x will be used.

P2P Network	Unique IPs	Port
Gnutella (8/2007)	377,246	6346
eDonkey (10/2007)	80,728	411
DirectConnect (10/2007)	175,139	4662

Table 1. Data sources and unique IP counts. Note that the given number of unique IPs is the number of IPs used for the generation of the statistical profiles. The actual networks maybe significantly larger.

Network Size (# IPs)	# SOAs
2^0 to 2^8 IPs	60,921
2^8 to 2^{16} IPs	14,577
2^{16} to 2^{24} IPs	1,296
2^{24} to 2^{32} IPs	22
Total	76,816

Table 2. Categorization of DNS SOAs by the size of the IP space that the SOA is responsible for.

3.1 Scanning DNS

Using GNU adns, we determined an approximation of the SOAs for all IP addresses. The algorithm started with all 255 networks of size 2^{24}. For each network, the code would first request the SOA for the first and last IP address in the network. If the SOAs were identical, the heuristic would assume that the entire range was under control of the particular SOA. If the SOAs were different, the network would be split into 255 subnets which would again be subjected to the same process.

Table 2 lists the number of IP addresses that each authority is responsible for (according to the above heuristic). For the statistics in Table 2, we used the full hostname of the DNS server to identify the organization (in other words, `ns1.example.org` and `ns2.example.org` would be treated as two different organizations).

3.2 Predicted Discovery Efficiency

Using the list of IPs for a peer-to-peer network and the break down of the IPv4 address space into domains by the SOA for each IP address, it is possible to determine how many peers are active in each domain. The resulting statistical profile is likely to differ between peer-to-peer networks; different networks appeal to different groups, for example, some peer-to-peer applications may have clients that are only available in certain languages. Similarly, support groups for particular networks also operate in a social and cultural context. This bias is not a problem for the proposed approach; in fact, the proposed approach works better because of this bias which is reflected in particularly high and particularly low probabilities for different organizations. However, this bias also means that statistical profiles must be created for each peer-to-peer application.

Table 3 provides a list of SOAs, the number of IPs for which the DNS server is the authority and the number of Gnutella peers falling into that range taken from the largest snaphots in our sets. The most stunning result is that at the time of the snapshot, almost 6% of the IPs in two organizations run Gnutella peers. Consequently, a peer scanning these organizations would be expected to succeed after an average of only 17 attempts. Given the size of the snapshot, a scan that would be oblivious to organizational bias would be expected to take on average 1,250 attempts.

This improvement in the number of peers that need to be probed is not realistic in practice. The reason is that achieving this kind of performance assumes that the network characteristics do not change over time, that current DNS information is available for free for the peer, and that the peer only scans the most promising organization. However, in order to repair network splits and to achieve the desired decentralization and its load balancing benefits, any heuristic must choose a trade-off between scanning highly promising organizations and scanning a broad range of organizations. The heuristic described in Section 2 will eventually return all IP addresses that have a probability higher than n^{-1}. For our experiments, we use $n = 1024$, ensuring that even in the worst case the probability of a single probe is still slightly better than a brute-force scan while also distributing the load among a broad range of organizations and IP addresses.

3.3 Observed Discovery Efficiency

The bootstrapping peers were provided with statistical information generated from that profile. The sizes of the generated statistical profiles, including full SOA names, IP ranges and respective probabilities, are given in Table 4. SOAs where the probability of finding a peer (based on the IP statistics available) is zero are not included in the database. Using this index, IP addresses were generated according to the heuristic presented in Section 2. The code then attempted to establish a TCP connection on the default port for the respective P2P protocol. The experiment considered a peer to be running a peer if the TCP connection was established successfully.

Since SOA range information was included in the database, no DNS requests were performed in the final experiment. In practice, an implementation would perform DNS queries to keep the SOA database up-to-date. The amount of DNS queries required corresponds to the frequency at which new DNS SOAs are created; we expect the necessary traffic to be insignificant, especially since the algorithm would tolerate somewhat outdated SOA information.

Table 4 also lists the average number of connection attempts needed to discover a peer. The "random global scan" does not use any statistical profiling data and just generates random IP addresses. The four "biased" approaches use (portions of) the hostname of the SOAs to map IP addresses to organizations. For example, "biased using TLD only" considers only the top-level domain as the "organization"; in other words, all IPs in the UK would be part of the same organization. Finally, "recent hostlist" uses random IPs from a list of IP addresses that is only a few months old (representing a common approach used today).

Table 4 shows the average number of IP probes required to discover a single peer over 50 runs; however, due to the randomized algorithm and the structure of the statistical profile, the variance is quite high. Depending on the P2P network, biasing the scan towards certain organizations improves the performance of random probing by a factor of 2 to 105. Unsurprisingly, the data also shows that using a sufficiently recent hostlist can produce connections with fewer probes. However, the results for E2DK are surprising, both in terms of how well the biased scan performs and in terms of how terribe a hostlist (which in this case is not even four months old) performs. This may in fact indicate that a stable core, i.e., long lived peers, for E2DK is relatively small. The smaller the stable core for a P2P network, the less useful a hostlist remains over time.

In all cases, the number of probes could be acceptable for an actual implementation, and as mentioned before, shipping a database with specific IP addresses raises various security and privacy concerns which do not apply to the statistical profiles.

While the presented experimental data is for IPv4, the overall size of the IP address space should not matter, as long as SOAs are not assigned to large amounts of unused address space. In contrast, the size of the peer-to-peer network in relation to the overall size of the Internet obviously still matters. However, small peer-to-peer networks can generally use hostlists – the costs of operating such a

Organization (SOA)	# IPs	# Peers
ns.pc-network.ro	254	15 (5.91%)
ns1.netplanet.ro	254	12 (4.72%)
ns.rdstm.ro	11,244	517 (4.60%)
...		
ns-a.bbtec.net	10,829,308	4 (0.00%)
rev1.kornet.net	10,857,115	1 (0.00%)
Total	2^{32}	3,741,099 (0.09%)

Table 3. Frequency of Gnutella peers in various domains taken from the largest snapshots in our sets. The table lists the three most dense domains, the least dense domain and the average density.

P2P Network	Gnutella		E2DK		DirectConnect	
	Probes	DB size	Probes	DB size	Probes	DB size
Random global scan	2425 ± 3089	0K	1875 ± 1780	0K	3117 ± 3080	0K
Biased, TLD only	833 ± 897	96K	18 ± 43	32K	1252 ± 1874	38K
Biased, domainname	1150 ± 1181	123K	74 ± 86	42K	623 ± 1599	52K
Biased, subdomain	849 ± 820	136K	56 ± 71	47K	1786 ± 2545	58K
Biased, FQN	817 ± 856	158K	51 ± 92	50K	1397 ± 2320	60K
Recent hostlist	245 ± 245	14964K	7039 ± 7185	320K	217 ± 211	712K

Table 4. Success statistics (average number of probes needed to find an open port and std. dev.) and compressed database sizes (in kilobytes) for various P2P networks.

centralized service for a small network would be insignificant and the likelihood of attention by powerful adversaries should be low.

4 Conclusion

By considering the geographic and organizational bias in the distribution of IP addresses participating in peer-to-peer networks, it is possible to construct a biased global address space scan that can efficiently bootstrap sufficiently large peer-to-peer networks. The main requirements for this method of peer-to-peer bootstrapping are that most peers use a default port and that the developers are able to obtain a list of IP addresses for the network; peer-to-peer networks usually grow over time, so it can be expected that by the time that centralized solutions become problematic developers will have access to such a list.

While the new approach has a clear advantage in terms of decentralization and elimination of critical points of failure, it cannot be expected to outperform the distribution of recent hostlists with the software in terms of the number of probes required. In particular, by providing a recent crawl of a given peer-to-peer network and using it as a hostlist, one can ensure with high probability that new peer can bootstrap into the system with minimal probing.

Acknowledgements

The authors thank David Barksdale, Nils Durner, Reze Rejaie and Daniel Stutzbach for providing us with a list of IP addresses for the various P2P networks. We also thank Hamid Hanifi for support with running the experiments.

References

[1] Clip2 DSS. Gnutella protocol specification v0.4. http://www.limewire.com, 2007.

[2] A. Gish, Y. Shavitt, and T. Tankel. Geographical statistics and characteristics of p2p query strings. In *6th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, Feb 2007.

[3] Saikat Guha, Neil Daswani, and Ravi Jain. An experimental study of the skype peer-to-peer voip system. In *The 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, Feb 2007.

[4] Daniel Stutzbach, Reze Rejaie, and Subhabrata Sen. Characterizing unstructured overlay topologies in modern p2p file-sharing systems. *ACM Transactions on Networking*, 2007.