

World's First Class C Web Census: The First Step in a Complete Census of the Web

Darcy Benoit

Jodrey School of Computer Science, Acadia University, Wolfville, Nova Scotia, Canada
 Email: darcy.benoit@acadiau.ca

Devin Slauenwhite, Nick Schofield and André Trudel

Jodrey School of Computer Science, Acadia University, Wolfville, Nova Scotia, Canada
 Email: {056012s, 049475s, andre.trudel}@acadiau.ca

Abstract — Our research goal is to measure the exact size of the World Wide Web (i.e., a census). The measure we are interested in is the number of publicly accessible web servers on port 80. We present the results from an initial survey of 3.5% of all addressable IP addresses. This is the largest survey of the Web performed to date. We then present results from the world's first census of all Class C IP addresses. We also describe our approach for a full census of the Internet.

Index Terms—World Wide Web, census, survey, distributed data collation, number of web servers on the Internet

1. INTRODUCTION

In the past 15 years, the World Wide Web (Web) has exploded beyond anyone's wildest dreams. Many people still claim that the Web is growing at a huge rate. But is it? How big is the Web today? Before this question can be answered, we must decide on a measurement. One possibility is to measure the number of pages or the size of the pages on the Web. Since search engines such as Google are not capable of spidering and cataloguing the entire Web, the best we can do with this measure is determine an estimate of the size of the Web.

If we wish to get an exact size of the Web, we must use a feasible measurement, such as counting the number of web servers. Specifically, the number of publicly accessible web servers available on the Internet on port 80. Using this measure, we have a fixed upper bound for the number of web servers: the number of IP addresses.

The IPv4 addressing system yields over 4 billion possible addresses:

$$\{0 - 255\}.\{0 - 255\}.\{0 - 255\}.\{0 - 255\}$$

$$= 256 * 256 * 256 * 256$$

$$= 4,294,967,296$$

For various reasons (such as IP addresses reserved for private networks), not all of the 4.3 billion addresses are useable. Of these, 3,702,423,846 could potentially host a web server.

Obviously, not all 3.7 billion IP addresses host a web server. But how many do? If we had one computer that ran 24/7 and averaged one IP address per minute, that computer would take approximately 7,000 years to complete a web census. This amount of time is clearly unreasonable. In order to bypass this obstacle, researchers have used estimation techniques to estimate the size of Web. A summary of these approaches appears in the next section. Can we do better than an estimate? We believe the answer is yes! We hypothesize that advances in hardware and programming languages, along with the power of the Internet and grid computing, that an exact measurement (i.e., a census) can be completed in a reasonable period of time. In this paper, we describe a survey of 3.5% of accessible IP addresses, a census of Class C addresses (14.5% of accessible IP addresses) and our plans to perform a first-ever census of the Web.

2. PREVIOUS APPROACHES: ESTIMATES

Many diverse approaches have been used during the past decade to estimate the size of the Web. In 1995, Hahn and Stout [9] released the second edition of The Internet Yellow Pages. They attempted to list and categorize the entire Web as it existed at that time in a printed format.

Lawrence and Giles [4][5] used the overlap between search engines to estimate the number of pages on the

Web that are available for search engines to find and catalogue. In another study, Giles and Lawrence [6] sampled the population of the Web by testing 3.6 million IP addresses, less than 0.1% of accessible addresses.

The Netcraft survey [10] sends HTTP queries to domain names from the DNS database.

The OCLC project [11] randomly generated IP addresses and then visited them to check for web sites. The OCLC project randomly visited 0.1% of accessible IP addresses, with results being based on the websites found at those IP addresses. The OCLC project ran from 1997 to 2002.

Population estimation techniques from biology were used in [12] to estimate the size of the web. In ecology, one of the oldest techniques for counting plants or animals in an area of known size is to divide the area into equal sized blocks called quadrats. The quadrats are then visited and the organisms of interest are counted. In [12], each quadrat consisted of 100 randomly generated IP addresses. We define a quadrat differently. An IP address consists of four octets. To create a quadrat, we randomly generate the first three octets, and allow the last octet to range between 0 and 255. For example, a valid quadrat would contain all the IP addresses between 125.210.098.0 and 125.210.098.255. Hence, a quadrat contains 256 consecutive IP addresses.

Details and drawbacks with each of the above estimation techniques are summarized in [12]. Regardless of their relative accuracy, they only provide estimates. We believe we can do better than this.

3. CURRENT APPROACH: CENSUS

As explained in the introduction, it would take one computer 7,000 years to perform a census of the Web when averaging one IP per minute. This is clearly an unreasonable amount of time. In order to speed up the census, we need to reduce the average time spent per IP address. There are two main factors that affect the average IP address visit time:

- Time out value: the amount of time spent waiting for a response at an IP address before determining that no response is forthcoming and that the IP address should be marked as having no web server.
- Parallelism: the number of concurrent requests a single machine makes and the number of machines making requests.

When we send a request to an IP address, we are inquiring about the presence of a web server and asking for the root page. What if there is no web server on that computer? What if no computer exists at that IP

address? We must determine how long to wait before determining that no response is forthcoming. How long do we wait? Using the example from the introduction, waiting one minute per IP address will result in a census time of 7000 years. If we can cut the wait time in half from one minute to 30 seconds, we can also cut the census time in half. Minimizing the time out value reduces the overall average time spent per IP address. But, when reducing the time out value there is a tradeoff between speed and accuracy. If we choose a timeout value that is too small, we will miss web servers that are unable to respond to our request within the timeout period. This can happen when a network connection is unusually slow and the timeout value is small. Increasing the time out value will increase the accuracy of the census by allowing web servers more time to respond, but will also increase the total time the census will take to run. We need to find a balance between speed and accuracy.

In order to determine a reasonable timeout value, we have run several experiments. We identified universities and non-educational websites that are geographically distant from our university (e.g., China, Australia, and Russia). For each distant website, we determined the IP address of its homepage. For each IP address, we generated the quadrat containing that address. For each quadrat, we visited each of its IP addresses and checked for a web server. Initially, we used a large timeout value (e.g., 1 minute). We then re-visited the quadrats with lower timeout values. We continued to lower the timeout value until we observed a reduction in the total number of web servers found, indicating missed web servers. In the first version of the web census software, we observed a loss of web servers when the timeout value dropped below 30 seconds. We preferred accuracy at the expense of speed and decided on a conservative timeout value of 30 seconds for our first survey. It is important to note that testing the timeout limit involves both university and non-university web servers. Since universities may have significantly better internet resources than the average website, we want to make sure that our experiment is not negatively affected by abnormally good internet access. Further fine-tuning of the algorithm should hopefully allow us to reduce the timeout.

The timeout value is important in reducing the web census time because a large proportion of the IP addresses visited will not have a computer attached to them. The present number of personal computers in use was estimated to be 804 million in 2004, with a projected estimate of 1 billion by 2007 [2]. Assuming that each of these machines is connected to the Internet, only 25% of the possible IP addresses are

likely to be assigned to a computer. If it were possible to determine the presence of a computer without having to wait for a timeout, we could speed up the census process by removing that timeout wait for 75% of IP addresses.

One approach we considered was the use of the “ping” network utility to check for the presence or absence of a computer at a particular IP address. Unfortunately, it was quickly discovered that “ping” was not suitable for the task at hand. Due to security issues and denial of service attacks known as “smurf” attacks, the CERT Coordination Center has advised that machines should be configured not to respond to ping requests [1]. As a result, many servers are refusing to respond to ping requests and many firewalls are filtering ping requests. Web servers known to be running and accepting connections, such as www.acadiau.ca, do not respond to ping requests. Given that firewalls are able to block requests to ports in the same way that they are able to block ping requests, the only way for us to check for the presence of a web server is to actually make a request on port 80.

The second factor affecting the average amount of time spent per IP address is the degree of parallelism that can be introduced in the census process.

Obviously the purely sequential model of one computer visiting one IP address at a time is unsuitable for a census. In order to reduce the amount of time needed to complete a census, we must both increase the number of concurrent requests a single machine makes and increase the number of machines making requests. For example, (assuming a timeout of 60 seconds) increasing the number of concurrent requests from one to 100 would reduce the census time from 7000 years to 70 years. If we increase the number of machines from one to 100, we would achieve the same decrease in census time from 7000 years to 70 years. Increasing the number of concurrent threads to 100 and increasing the number of machines to 100 results in reducing the web census time from 7000 years to 257 days. We can easily see how the combination of concurrent requests combined with multiple machines quickly reduces the amount of time needed to census the Web. The amount of time estimated to census the web, measured in months and assuming a timeout of 60 seconds and machines running 24/7 can be seen in Table 1. For example, 1000 computers each running 50 concurrent requests would terminate the census in 1.7 months. Time estimates for a timeout of 30 seconds under the same conditions can be found in Table 2.

Table 1. Time estimates to census the Web, in months (unless otherwise noted), with machines running 24/7 and a connection timeout of 60 seconds.

Requests	Machines					
	1	50	100	1000	2000	
1	85704.3	1714.1	857.0	85.7	42.9	
10	8570.4	171.4	85.7	8.6	4.3	
25	3428.2	68.6	34.3	3.4	1.7	
50	1714.1	34.3	17.1	1.7	0.9	
100	857.0	17.1	8.6	0.9	0.4	
1000	85.7	1.7	0.9	0.1	(1.3 days)	

Table 2. Time estimates to census the Web, in months (unless otherwise noted), with machines running 24/7 and a connection timeout of 30 seconds.

Requests	Machines					
	1	50	100	1000	2000	
1	42852.1	857.0	428.5	42.9	21.4	
10	4285.2	85.7	42.9	4.3	2.1	
25	1714.1	34.3	17.1	1.7	0.9	
50	857.0	17.1	8.6	0.9	0.4	
100	428.5	8.6	4.3	0.4	0.2	
1000	42.9	0.9	0.4	(1.3 days)	(0.6 days)	

It should be noted that previous surveys of the Web involved both slower computers and networks. The increase in computing power over the past several years has made it feasible to run a Web census program in the background of a regular laptop computer without serious negative effects to a regular user. This updated computing power, combined with faster local networks and general increased bandwidth at the Internet backbone level provide the ability to substantially increase the number of requests individual machines can make.

4. IMPLEMENTATION: CLIENT-SERVER GRID COMPUTING

A. Our Approach

The proposed census model is a large grid project where distributed clients perform the actual census work. The client computers connect to a centralized database server in order to obtain information about IP ranges that they should check for Web servers. The centralized server will also be responsible for receiving information from each of the clients on websites that they have visited. Similar projects such as grid.org (cancer research) [8], SETI@home (analyzing radio telescope data) [13] and distributed.net (cryptographic problems) [3] have all used distributed computing to work on problems too large for a single computer. Our project uses the same basic structure where client machines do the work and a centralized server collects and analyzes the data.

B. The Client

The client is written in Java. A write-once, run-anywhere implementation will allow us to easily run the software on both Windows and Linux/Unix machines, both of which are available for the project. We expect each machine to be able to visit and store information for a large number of IP addresses, so each client will use an embedded database management system (DBMS) called Derby to manage data on the client machine. Derby will provide a level of security to the data on the client machine, using transactions to ensure that data is correctly written to the disk, tracking unfinished data when a client machine is shut down unexpectedly, and allowing the client to query the collected data and recover from shutdowns appropriately. The type of data stored includes quadrats assigned by the server that need to be surveyed, the IP addresses visited, the date of the visit, the presence or absence of a Web server, and (if the web server is present) the return code and the root webpage of the Web server.

Note that a database is used because the access patterns and volume of data involved makes the use of a flat file infeasible. The speed at which individual computers will be querying the web will result in tens of thousands of IP addresses to be tracked per machine. Each client will run multiple threads, and each will need to be able to write data to the database concurrently. Using a DBMS will help solve the concurrent data access problem.

The client machine is responsible for collecting data on a number of quadrats over some period of time. The amount of time needed to collect the data will depend on the number of threads that the client is able to run and the number of quadrats passed out by the server. It is important that each client be assigned many quadrats and that the IP addresses within a quadrat not be visited in a consecutive fashion. We do not want to arouse the suspicion of system administrators worldwide by incrementally scanning IP addresses in a network.

C. The Server

With 3.7 billion IP addresses to be visited during a Web census, it is not practical to store bits and pieces of Web census information across numerous client computers. Our implementation calls for a DBMS server that will act as a central repository for the census data. The server is a Windows Server 2003 machine what will run IBM's DB2 database management system. Each client machine, when finished their assigned quadrats, connects to the central server. The client machine transfers all of the census data to the database server in compressed XML format. Once the server has received the data from the client, the client machine is able to purge the information from its own Derby database and insert a new set of quadrats generated by the server. It should be noted that the client machine keeps a copy of the compressed XML file until the next submission to the server. This acts as a mini-backup in case the compressed XML file is corrupted during transmission to the server. Data stored by the server will then be available for analysis.

Several steps are taken to ensure the integrity of the data. Data is tracked according to the machine that collected the information. This allows us to group data based on who retrieved the data from the Web. If a problem is associated with data from a particular user, we can remove that data from the database and assign those quadrats to another client machine. Also, if a client machine has some failure during the census (e.g., failed hardware, disconnected from the Internet for too long, etc.), the server machine is able to identify the quadrats assigned to

that client and reassign those quadrats to another client.

When assigning quadrats to clients, the server generates the quadrats randomly. The randomly generated quadrats are then checked against the list of quadrats already visited or currently assigned to another client. Duplicate quadrats are discarded and, if needed, new quadrats are generated for the client. There are two reasons for doing this. The first is that it spreads the IP address probes across the internet, and does not concentrate them within a single local network. The second reason is that if for whatever reason it is impossible to complete the census, we can use the data collected and techniques similar to the ones used in [12] to estimate the size of the Web.

5. PRELIMINARY RESULTS – OUR FIRST SURVEY

A. Initial Results

A preliminary test of our system was started on September 20th, 2005 and ran until February 15th, 2006. The client software was placed on nine desktop computers, all running Novell’s SuSE Linux. We were able to run 150 concurrent threads on Linux, which is a better throughput than on Windows XP. The machines used were located in a research lab, allowing us to closely monitor the machines throughout the census. Some changes were made to the client software throughout the testing period in order to increase efficiency and robustness. Since these machines were used by other researchers, they did not run the census software on a 24/7 basis. Despite this fact, we were able to collect a substantial amount of data in the five months the system was running. Our results can be seen in Table 3. In the five months that the system was running, we were able to survey 3.5% of the accessible Internet, the largest web server survey to date.

Data was stored for each discovered web server in our server database, allowing us to query the data to answer several questions. One of the first queries we performed was to determine the distribution of web servers across the three classes of networks – A, B, and C. Table 4 shows the distribution of servers across the network classes. It should be noted that Class A contains 57% of all of the IP addresses on the Internet, and 46.1% of the web servers found. Class B makes up 29% of the addressable space, but only 11.4% of the web servers we found were in Class B networks. Finally, Class C networks account for only 14% of the IP addresses available, with 42.5% of the servers located in this address space. These numbers show that the distribution of web

servers is not equal across network classes, and that from the perspective of web servers, Class C is fairly densely populated, while Class B is underutilized.

B. Server Distribution

It was initially assumed that the distribution of servers within a quadrat would not be random. This assumption is based on the belief that the purpose of the network would determine the distribution of servers within its quadrats. For example, organizations may be likely to clump their web servers into some limited number of subnets due to router, firewall and networking issues as well as the physical proximity of the machines. It is unlikely that an organization would put a web server in the same subnet as dynamic user IP addresses.

In order to test the theory that web servers are not randomly allocated in a quadrat, we use a *negative binomial distribution* [7] to model our web server population. To determine if our data can be described by the negative binomial distribution, the *U-Statistic Goodness-of-Fit Test* is used. In order to use the negative binomial distribution, the U statistic cannot be more than twice the size of the standard error of U. Table 5 shows that each network fails the goodness-of-fit test. Unfortunately, this does not prove nor disprove that servers are clustered within quadrats since there are other statistical models for aggregated data which may be appropriate.

Table 3 - Results of web server survey.

Quadrats Scanned	506,212
IP Addresses Scanned	129,590,272
Servers Found	373,063

Table 4 - Web server distribution across network classes.

Network Class	IP Addresses scanned	Servers Found	% of Servers Found
A	73,226,496	173,167	46.1%
B	37,523,712	42,716	11.4%
C	18,840,064	159,720	42.5%

Table 5 - Negative Binomial U-Statistic Goodness-of-Fit Test

Network Class	U Statistic	Standard Error Of U	U statistic is X times Standard Error of U
Class A	35.461	0.4320	X = 82.1
Class B	17.826	0.3714	X = 48.0
Class C	-612.122	52.3392	X = 11.7

To rule out the possibility that our data is random, we used the “*Index of Dispersion Test*” [5] which determines the validity of using the Poisson distribution. The ideal dispersion for the Poisson distribution is 1.0. To check that the index of dispersion is close to 1.0, we must calculate chi-squared and standard normal deviate values. If the distribution pattern is random then the value of z , the Standard Normal Deviate, should be between -1.96 and 1.96 [5]. A z value below -1.96 represents a spatial pattern of uniform distribution and a value above 1.96 represents a spatial pattern of aggregated distribution. The standard normal deviate values for each network class are shown in Table 6. Since the standard normal deviate value for all network classes is significantly larger than 1.96, our data is aggregated, but the negative binomial technique cannot be used to model our population. Other distribution models will be considered in the future.

We determine a population estimate from our data by multiplying the observed mean for each quadrat by the total number of quadrats in that network class. Table 7 shows the different population estimates for each network class. This gives us an estimate of 10,731,148 web servers connected to the internet. It should be noted that although our survey visited many more IP addresses than previous estimation attempts, our population estimate is significantly lower than other estimates. Comparisons of web size estimates can be found in Figure 1. Possible explanations for the difference in estimates are sample size and model used, and the dynamic nature of the Web.

Table 6 - Index of Dispersion Test

Network Class	Standard Normal Deviate (z)
Class A	6678
Class B	4765
Class C	3261

Table 7 - Network class population estimates.

Network Class	Average Number of Servers per Quadrat	Population Estimate
A	0.6054	4,959,437
B	0.2914	1,221,027
C	2.1702	4,550,684

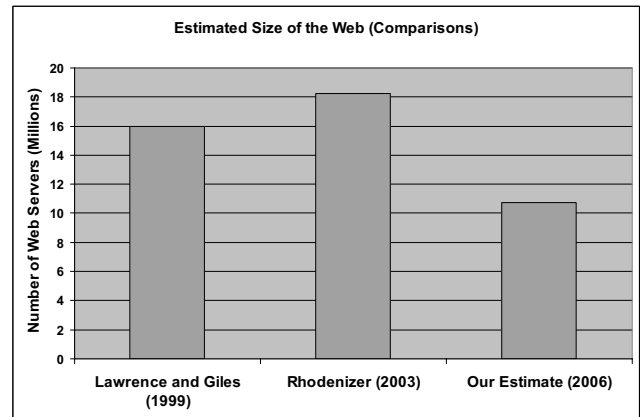


Figure 1 - Comparison of web size estimates.

6. OUR NEW APPROACH – A CLASS C CENSUS

A. Issues with our first survey

After the completion of our first survey, several issues associated with the software and the way that we accessed the web servers came to light. While the software worked well to retrieve the data, it was not as robust as needed and required a rewrite. Although the quadrats were randomly generated, the IP addresses within a quadrat were accessed sequentially. This caused some network administrators to assume that we were attacking their network. We needed to change the way in which we scanned IP addresses in order to avoid being blocked from networks by company firewalls. We also wanted to add some other features to our code to increase control and usability. As a result, a second version of the retrieval system was implemented. The new application was used to perform a complete census on all IP addresses in the Class C address space.

B. Our new client

The new version of our client application required a complete rewrite of the code. Memory usage was an issue, so the implementation was adjusted until main memory usage was approximately 128MB, including the JVM and Derby database. The client switched from using a Java http library to using straight sockets to connect to the web servers, avoiding some issues with the http library. Throughput was a huge concern, so the client was written to handle larger numbers of threads than used in the previous survey.

One implementation concern was associated with the order in which we scanned IP addresses within a quadrat. Recall that to create a quadrat, we randomly generate the first three octets, and allow the last octet

to range between 0 and 255. For example, a valid quadrat would contain all the IP addresses between 125.210.098.0 and 125.210.098.255. To scan this quadrat, we visit each IP address incrementally beginning with 125.210.098.0. If the client has a 100 thread count limit and a 30 second timeout value, then the addresses 125.210.098.{0-99} are accessed simultaneously. Within 30 seconds, the next 100 are accessed. This brute-force hit on individual networks triggered emails from various system administrators and automated firewall software.

In order to avoid the above illusion that we are performing an intrusive port scan on a network, we changed the order in which quadrats are generated and visited. Quadrats are generated in blocks of size 222. Assume a quadrat consists of 3 octets of the form A.B.C. The value of A is set to 1 and the values of B and C are set to 0, giving us the quadrat 1.0.0. The value of A is incremented from 1 to 222, providing 222 quadrats in the form of 1.0.0, 2.0.0, 3.0.0, ..., 222.0.0. Note that all IP addresses between 223.0.0.0 and 255.255.255.255 are reserved so they are not checked, leaving us to search the range {1-222}.{0-255}.{0-255}. Once the server has passed out the first 222 quadrats, the value of A is set back to 1 and the value of B is incremented by 1. The server will then pass out the quadrats 1.1.0, 2.1.0, 3.1.0, ..., 222.1.0. This pattern will continue until the value of B is 255. When 222.255.0 is reached, A will reset to 1, B to 0 and C to 1. The server will then begin to pass out quadrats starting at 1.0.1, 2.0.1, 3.0.1.1, etc.

Using the above order for generating quadrats, each client receives a group of 2000 quadrats. In order to span as many networks as possible when checking for servers, the client will generate a random number between 0 and 255. Assume that the number 44 is generated. Each of the 2000 quadrats will be checked, in order, to see if a server exists in the quadrat at number 44. For example, if 1.2.1 and 2.2.1 are assigned to the client, then we check both 1.2.1.44 and 2.2.1.44 for a server. Once all 2000 quadrats are checked, a new number between 0 and 255 is generated (44 excluded) and all 2000 quadrats are checked for a server at that position. This pattern is continued until all numbers between 0 and 255 are checked. This scanning pattern was chosen because it spreads out the queries across many different networks.

The client also features a new autonomic thread management system that allows the software to manage its number of threads. After a set number of quadrats have been scanned by the client, it will enter into a validation mode. In the validation mode, 10

quadrats are chosen from the new set of quadrats assigned by the server. Those quadrats are then scanned at a thread count of 150 threads with a timeout value of 30 seconds. This is then considered our baseline, telling us the number of servers in the selected quadrats. The client then rechecks the same quadrats using the current thread count (initial value of 300) and a timeout set by the server. If the number of servers differs by less than 1%, then the number of threads used by the client is adjusted upwards and the validation is run again. This continues until the error rate is above 1%, and then the previous thread count value is used. The clients are capped at a 100 thread count increase per validation cycle to avoid excessive fluctuation. The client can also decrease the thread count in order to get an error that is below 1%. There is no cap on thread decreases per validation cycle. The error rate was chosen as a fair compromise between data validity and the census completion timeframe.

Our new implementation uses web services for communication between the client and the server machine. Quadrats are passed from the server to the clients to check, and completed searches are passed from the client to the server in a compressed XML file. The server is responsible for uncompressing and parsing the XML file, inserting the data into the database, and passing out new quadrats to the client to be scanned. It should be noted that the upgraded version of the server in our census of Class C provides methods to control the client software. For example, the server software allows us to set the number of quadrats passed out to each client, the maximum number of threads each client can have, how often each client goes through the autonomic thread setting process, how long quadrats can be absent before they "time out" and are passed out to another client, and the general timeout value for the census. This information is passed to the clients via web services. The machine used for the server in the Class C census is the same hardware as the server in the first survey, but Windows Server 2003 was replaced with CentOS in order to facilitate server access and to allow the server to act as a client. DB2 Express C is the DBMS used.

7. CLASS C CENSUS RESULTS

A. Overall Results

The first real test of our new system was a census of all IP addresses in the Class C address space. Class C ranges from 192.0.0.0 to 222.255.255.255. We had a peak of 18 machines working on the Class C

survey, although the number of machines varied during the course of the survey. The survey was started on July 4th, 2006 and was completed on July 27th, 2006, for a total time of 23 days. Due to testing results from the new client, we were able to reduce the timeout value for the survey from 30 second to 20 seconds. We predict that our current setup would allow us to census the entire Internet in approximately 5.5 months. We collected information on address distribution, mean server density, quadrat distribution per IP address, operating system type, web server type and web server response code. Our census of Class C found 6,908,877 servers, with an average of 3.2944 servers per quadrat. Results for our census of Class C can be found in Table 8.

In our first set of experiments, our client code only collected data from web servers that returned 100 and 200 level response codes, plus response codes 301 and 302. This ignored all response codes in the 400 and above ranges, most of which are errors. It was decided that the Class C census should be more inclusive of all web servers, so all server responses were included.

In order to accurately compare the results of our July 2006 survey with the fall 2005 survey, we need to only consider the Class C data from the first survey and we must take into account the extra web servers counted in the second survey. Only 4,518,347 servers from the second survey had response codes that would have been counted in the first survey. When comparing the results of the two surveys, the first survey estimated an average of 2.1703 servers per quadrat. The second survey of Class C counted 2.1545 servers per quadrat.

B. Server Software and Operating System

The census software collected information concerning the type of web server used and the type of operating system used by the web server machine. Since these values are self-reported by the web server when responding to a request, some web servers chose to not report this information. As such, our observations of web servers and operating systems have categories of “empty” when a value was not returned. We also have a category of “other” which includes all of the operating systems or web servers not included in the following charts.

Table 8 - Class C census results.

Quadrats Scanned	2,097,152
IP Addresses Scanned	536,870,912
Servers Found	6,908,877
Average Number of Servers per Quadrat	3.3

Figure 2 shows a comparison of web server software. The calculation of web server software was fairly straightforward, with all versions of a piece of software included in the same count. For example, an Apache server run on a Windows, Linux or Unix platform was included in the Apache total.

Figure 3 shows a comparison of operating systems used for the web servers. In this case, the calculation of OS type was not so straightforward. We assumed that any server running IIS as their web server would have to be using the Windows OS, so we included all reports of IIS as the web server as Windows servers. Any other machine that reported the OS as “windows”, “win32” or “winnt” was also included in the Windows count. Linux machines were counted as those that self-reported the OS type as “linux”, “redhat”, “red hat”, “debian”, “ubuntu”, “slack”, “slackware”, “centos”, “mandrake”, “mandriva”, “suse”, “knoppix”, “fedora” or “gentoo”. The “unix” category was any machine that reported itself as “unix”, not including those that reported themselves as “linux”. If a machine reported themselves as “redhat unix”, then they would be counted under the “linux” category and not the “unix” category.

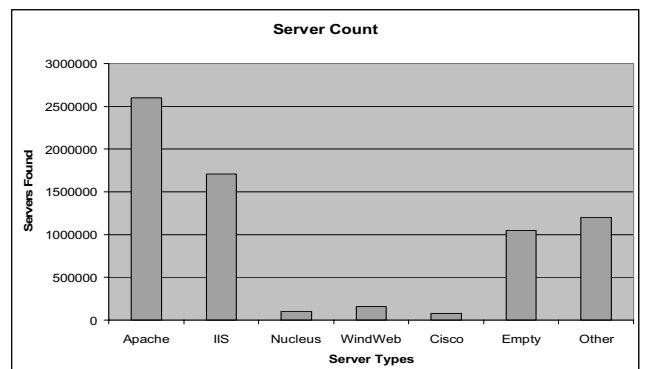


Figure 2 - Self-reported web server types.

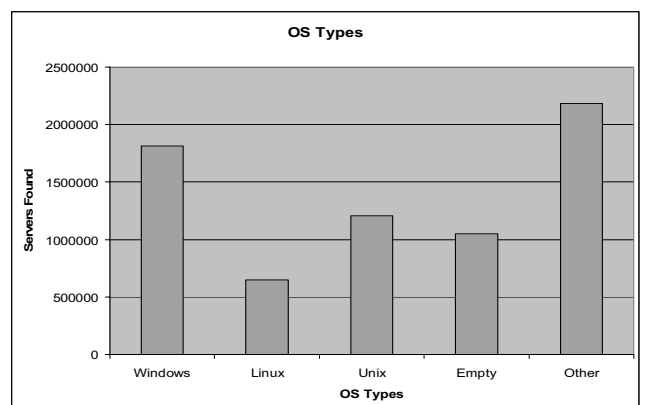


Figure 3 - Self-reported operating system types.

C. Web server distribution across networks

Another area of interest was the distribution of web servers across networks. The first question concerned the mean server density. The mean server density is the mean number of servers per quadrat when the quadrat contains one or more servers. Although the average number of servers per quadrat is 3.3, many quadrats are empty and were not included in the mean server density calculation. The mean server density is approximately 14 servers per quadrat for those quadrats with at least one server.

Aside from server distribution inside of the quadrats, we were interested in viewing the “big picture” server distribution across IP addresses. In order to see how web servers were distributed across the IP address space, we counted the number of web servers based on the first number of the IP address. Class C networks span from 192.0.0.0 to 223.255.255.255, so we have determined the total for each top-layer network. Figure 4 shows the distribution of servers across the address space. For example, the range of IP addresses from 205.0.0.0 through 205.255.255.255 contains roughly 100,000 web servers. The distribution of servers is not uniform over the address space. For example, the 192 address space has a fairly low number of servers. This is to be expected, as a portion of the 192 address space is reserved for private networks and cannot be used for public web servers. (192.168.x.x is reserved). The 214 and 215 address spaces are also

curiously low. 214.0.0.0 through 215.255.255.255 are owned by “DoD Network Information Center”, which appears to be owned by the US Department of Defense [14]. With such a large address space owned by a military organization, it is understandable that they have a low number of web servers accessible to the general public. Conversely, the 216 address space is split up over a number of companies and has a high number of web servers. Further investigation is required for 216.

D. Web server numbering distribution

One area of interest was the distribution of servers across the available server numbers in the last octet. Are servers randomly distributed across the address space, or is there some pattern to server number distribution? Figure 5 shows the distribution of servers across the available server numbers for all IP addresses in Class C. For example, the bar above 100 on the x-axis gives the total number of web servers found in the range {192-223}. {0-255}. {0-255}. 100. As we can see, the distribution does not seem to be random as there is a wide variation in the number of servers at individual numbers. There seems to be a spike in servers towards the middle of the graph, and servers with low values tend to occur in pairs. There is a general trend where the lower numbers tend to have more servers than the higher numbers. These casual observations require further study.

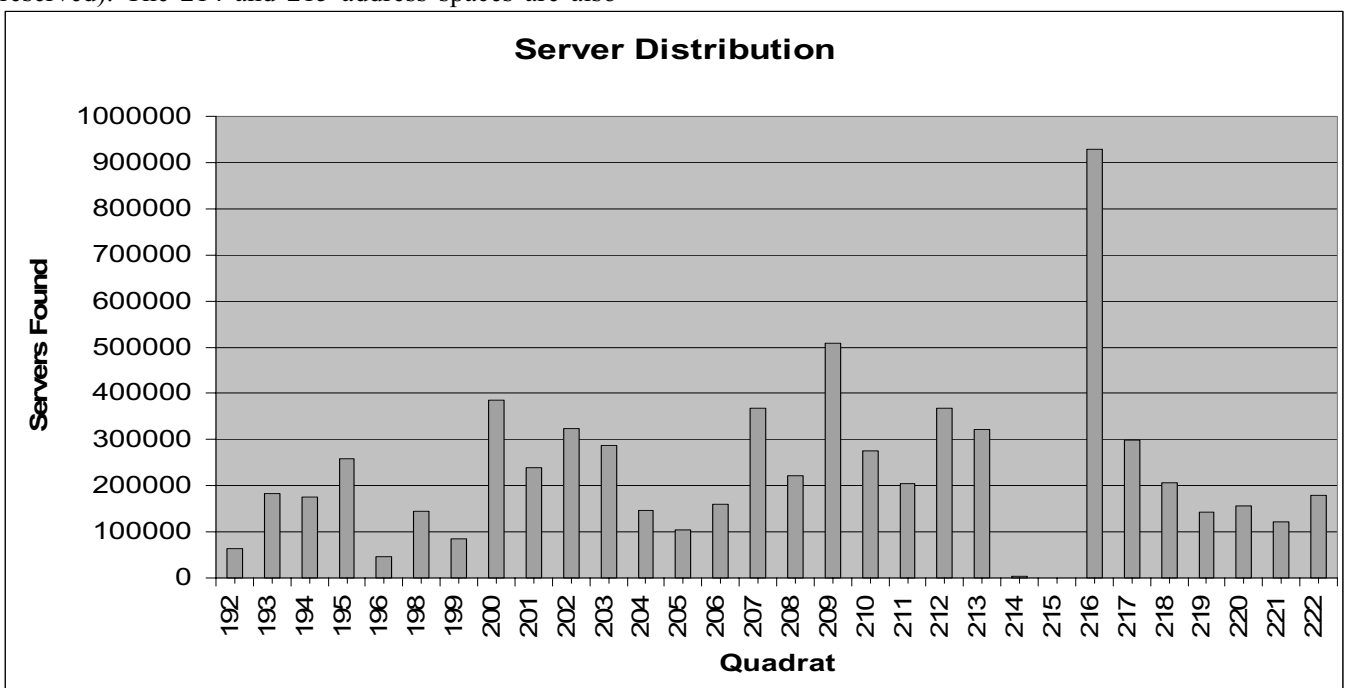


Figure 4 - Web server distribution across network address space.

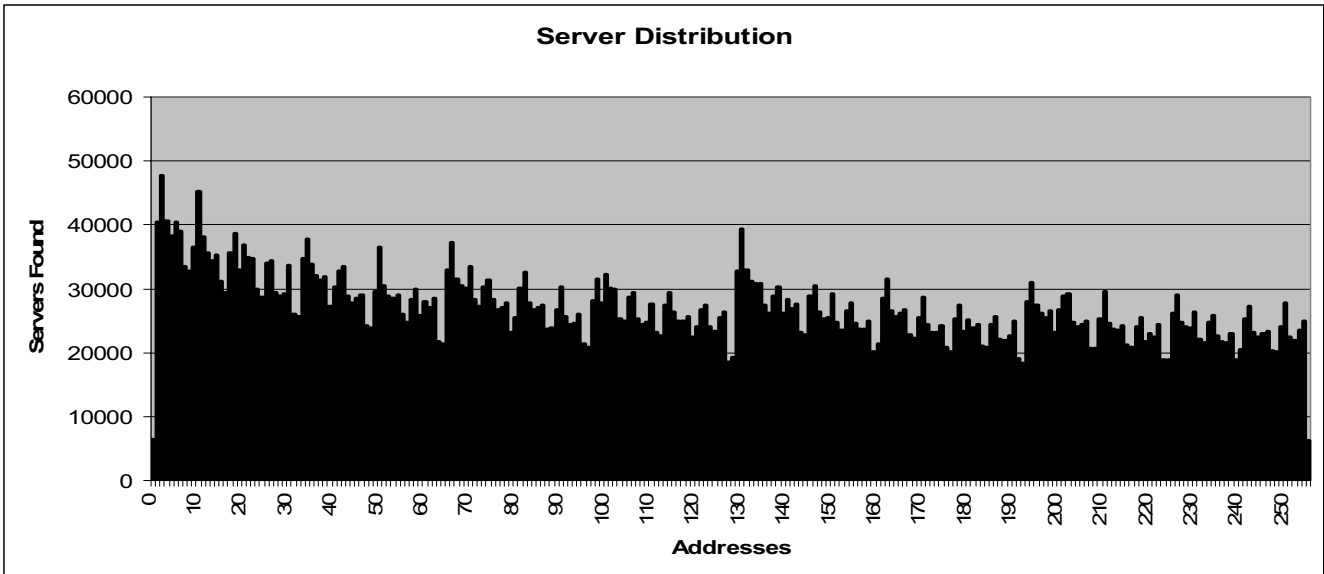


Figure 5 - Distribution of servers according to assigned IP number.

E. Number of Servers per Quadrat

Beyond the basic distribution of servers based on number, we are able to determine the number of servers per quadrat and the average size per “cluster” of servers. A cluster is a group of two or more consecutively numbered servers within a quadrat. The average cluster size within Class C is 5.39 servers per cluster. The maximum cluster size is 256. There were only five quadrats where all 256 addresses had a web server. The five full quadrats are 195.204.174, 200.147.102, 200.187.254, 202.242.174 and 216.219.174.

Figure 6 maps the number of quadrats based on the total number of servers within the quadrat (clustering is not considered for these totals). For example, 101,927 quadrats have only one server, while 11,692 quadrats have 10 servers. As expected, the number of quadrats decreased as the number of servers increased. The curve flattens beginning around 130 servers per quadrat and stays fairly flat until 240 servers per quadrat. Although the scale of Figure 6 makes it difficult to see, the curve increases slightly, with larger numbers of quadrats having 240+ servers. Figure 7 shows the quadrat count based on number of servers from 200 servers to 256 servers.

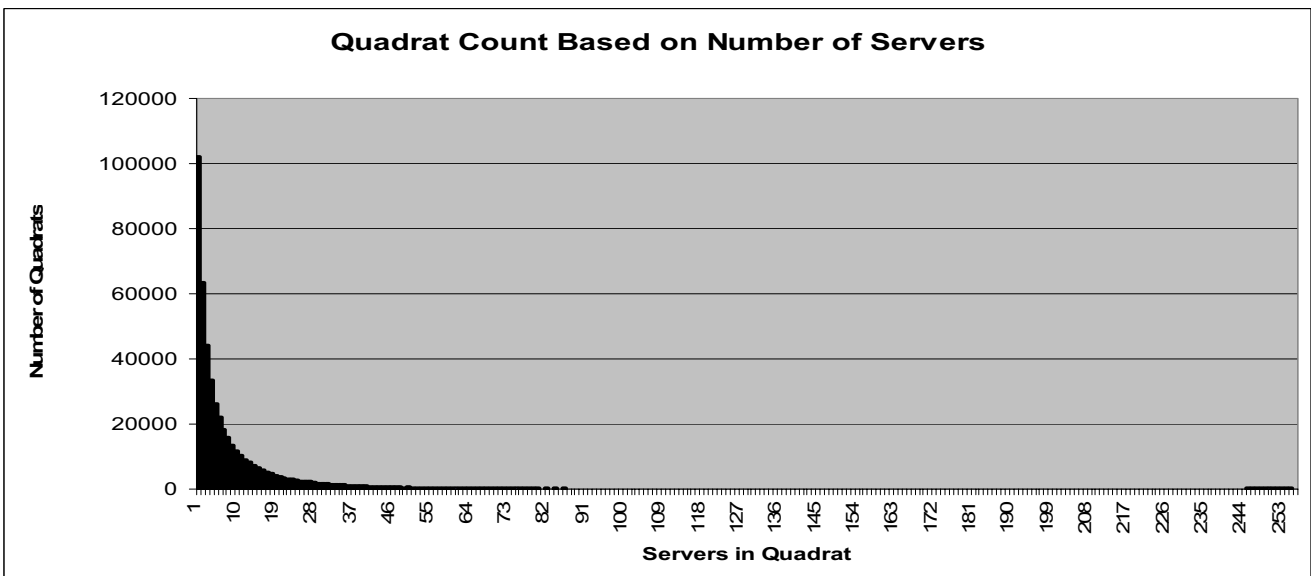


Figure 6 - Quadrat count based on number of servers.

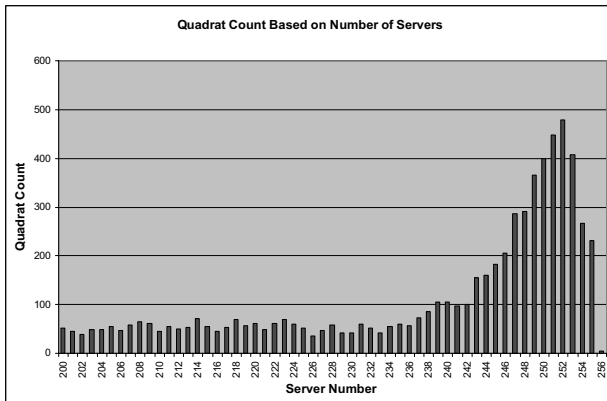


Figure 7 - Quadrat count from 200 to 256 servers.

8. CONCLUSIONS AND FUTURE WORK

A. Conclusions

The goal of our proposed research is to census the Internet in order to get an exact measure of the number of web servers on port 80. Note that we do not measure web servers on any machine behind a firewall that blocks or otherwise filters port 80 traffic. In addition to the number of servers, downloading the web page content during our census will allow us to measure the following:

- The number and percentage of sites using a particular language such as English, French, Chinese, etc.
- The number of sites dedicated to e-commerce, education, adult entertainment, etc.
- The market share of web hosting software.
- The average and largest size of consecutive clusters of IP addresses that host web servers.
- The distribution of web servers in a quadrat (i.e., Do web servers typically appear in one section of a quadrat?)

Collected information can be used in various ways. Site language information can be used to focus browser support for particular languages. The number of e-commerce sites is of interest to industry analysts. The distribution of web servers in a quadrat will impact web server security.

The major impact of our proposed research is that it will be the world's first Web census. Future research will involve re-doing the census at regular intervals (e.g., every 6 months) to measure the growth/decline of the Web. If successful, this will be the first time in the Web's history that we know its exact size and how it changes over time.

Our first survey allowed us to visit 3.5% of addressable IP addresses on the Internet. The data collected from this survey provided us with

information about web server distribution across network classes and a useful trial case for our software. While we still have not mined all of the useful information from that first survey, we pushed on with a new and improved version of the software, resulting in a full census of Class C, or 14.5% of the addressable Internet. This is the first full census of any single portion of the Internet, and the data retrieved from this census will be further studied at length. Information gained from this second survey will be applied to further fine-tune our data collection system, which will be further tested on a second census of Class C. It is expected that once the second survey of Class C is complete, a full census of the Internet will be attempted.

B. Future Work

After we completed our first survey in the fall/winter of 2005/2006, we firmly believed that a very large number of computers would be required to complete a census of the entire Web. Given the results that we have achieved with our census of Class C, we now see that the census of the Internet is feasible with a reasonable number of machines. Our current implementation running on 18 machines should be able to complete a census in about 4.5 months. This is significantly lower than our previous predictions and puts a web census firmly within our reach. In order to get the most accurate census results, the amount of time in which the census is completed needs to be shortened as much as possible. The most accurate results would occur if we were able to do a census in a few days rather than a few weeks. In order to achieve such a feat, we would have to distribute the client to a large number of users. Currently, the client software and the server are designed to handle large numbers of clients. What is needed is access to significant numbers of machines to be able to perform the work.

Acadia University runs a program that is known as the "Acadia Advantage". This program puts laptops in the hands of each and every student for the school year, and identical laptops in the hands of faculty year-round. This gives us a working base of approximately 4000 laptops with identical hardware and software specifications. It would be to our advantage to install our web census software on the disk image used for the laptops, giving us 4000 machines all doing census work at the same time. Unfortunately, there are several obstacles that we must overcome before we are able to utilize these machines.

The first obstacle is making the program robust enough to be able to run on a student's computer. We have mostly accomplished this task, only needing to better manage client upgrades to make them more user-friendly. The second obstacle is more of a problem. Currently, Windows XP machines with Service Pack 2 (SP2) do not allow more than 10 Internet connections at any given point in time. Since our client requires hundreds of waiting threads, Windows XP SP2 machines cannot be used to their full potential. We can cut the number of threads to five, allowing the client program to run in the background without interfering with the user. This greatly reduces the usefulness of the program, but given the large number of machines available, it might be a useful route to take.

Several of our students and faculty use versions of Linux on their laptops. Since Linux does not have the same impediment as Windows XP SP2, those machines would be able to achieve greater throughput. Installing the client on those Linux machine would be a great help to our project.

As we work in the direction of getting all of the machines on campus involved in this project, we have several other issues to consider. How do we know if the client is connected to the Internet? How do we know that the client's Internet connection is a "good" connection and not a bad one? Should we allow our client software to execute when the machine has a wireless connection? These concerns, and many others, will need to be addressed before we are able to roll out our client software en masse. Until that point in time, we will continue to run our high-volume dedicated census machines with Linux in order to collect the needed census data.

9. ACKNOWLEDGEMENTS

The first and last authors are supported by NSERC Discovery Grants.

10. REFERENCES

- [1] CERT Coordination Center, Advisory CA-1998-01 Smurf Denial-of-Service Attacks, <http://www.cert.org/advisories/CA-1998-01.html>
- [2] Computer Industry Almanac Inc., Press Release, 9 March 2005, Retrieved 12 July 2005 from <http://www.c-i-a.com/pr0305.htm>.
- [3] distributed.net, Retrieved 12 July 2005 from <http://www.distributed.net/>.
- [4] C.L. Giles, & S. Lawrence, "Searching the Web: General and Scientific Information Access", *IEEE Communications Magazine*, 1999, 116-122.
- [5] C.L. Giles, & S. Lawrence, "Searching the World Wide Web", *Science Magazine*, 280, 1998, 98-100.
- [6] C.L. Giles, & S. Lawrence, "Accessibility of Information on the Web", *Nature Science Journal*, 400, 1999, 107-109.
- [7] C.J. Krebs, *Ecological Methodology* (2nd Edition) (Menlo Park: Addison-Welsey Educational Publishers, 1999).
- [8] Grid.org, Retrieved 12 July 2005 from <http://www.grid.org/>.
- [9] H. Hahn, & R. Stout, *The Internet Yellow Pages*, 2nd Edition. (Berkeley: Osborne-McGraw-Hill, 1995).
- [10] Netcraft's Web Server Survey. Retrieved May 11, 2004 from http://news.netcraft.com/archives/web_server_survey.html.
- [11] Online Computer Library Centre (OCLC), Inc.'s Web Characterization Project. Retrieved May 11, 2004 from <http://wcp.oclc.org>.
- [12] D. Rhodenizer, & A. Trudel, "Estimating the size and content of the World Wide Web", *Proceedings of the Third IASTED International Conference on Communications, Internet, and Information Technology*, US Virgin Islands, 2004, 564-570.
- [13] SETI@home, Retrieved 12 July 2005 from <http://setiathome.ssl.berkeley.edu/>.
- [14] <http://www.arin.net/whois/> (Retrieved 31 July 2006)

Darcy G. Benoit is an assistant professor at the Jodrey School of Computer Science at Acadia University, Canada. He received his B.Sc from St. Francis Xavier University in Nova Scotia, Canada (1995), M.Sc. (1997) and Ph.D. (2003) from Queen's University at Kingston in Ontario, Canada. Dr. Benoit's areas of research include autonomic computing, distributed computing and database management system performance. Dr. Benoit is a member of the IEEE.

Devin Slauenwhite is a video game programmer at HB Studios in Lunenburg, Nova Scotia. He received his Bachelor of Computer Science with Honours from Acadia University in the spring of 2006. Mr. Slauenwhite completed the fall 2006 web survey as a part of his honours thesis work.

Nicholas Schofield is a student at Acadia University. He will receive his Bachelor of Computer Science (co-op) with a specialization in E-Commerce in the fall of 2006. Mr. Schofield completed the census of Class C as part of a summer work with Drs. Benoit and Trudel. Mr. Schofield will begin work at IBM Canada in Toronto in the fall of 2006.

André Trudel is a professor at the Jodrey School of Computer Science at Acadia University, Canada. He received his PhD from the University of Waterloo in Ontario, Canada. Dr. Trudel's research is in the area of artificial intelligence.