The Effects of Server Placement and Server Selection for Internet Services

Ryuji Somegawa *Kenjiro Cho* *Yuji Sekiya* *Suguru Yamaguchi*
WIDE Project
{*somegawa,kjc,sekiya,suguru*}@*wide.ad.jp*

Abstract

Many services on the Internet are provided by multiple identical servers in order to improve performance and robustness. The number, the location and the distribution of servers affect the performance and reliability of a service. The server placement is, however, often determined based on the empirical knowledge of the administrators. This paper investigates issues of the server placement in terms of the service performance and the server load.

We identify that a server selection mechanism plays an important role in server placement, and thus, evaluate different server selection algorithms. The result shows that it is essential to the robustness of a service to employ a mechanism which distributes service requests to the servers according to the measured response time of each server.

As a case study, we evaluate the server selection mechanisms employed by different DNS (Domain Name System) implementations. Then, we show the effects of the different server selection algorithms using root-server measurements taken at different locations around the world.

1 Introduction

As the Internet continues to grow at an explosive rate, the increasing number of services on the Internet become indispensable to our life. Many services on the Internet are provided by multiple identical servers in order to improve performance and robustness. For such services, server placement is an important factor of the quality of a service. Server placement has been a subject of research where the number, the location and the distribution of servers are studied so as to increase the total system performance and the reliability of the service.

Although the best-server selection is often assumed for server placement, we found it is not the case with many Internet services in use, notably with DNS (Domain Name System), and use of different selection mechanisms has a significant impact to server placement strategies. This paper investigates server selection mechanisms, and explores issues of server placement using different selection mechanisms. We categorize server selection algorithms and illustrate their behavior in simple synthetic situations.

As a case study, we evaluate the server selection mechanisms employed by different DNS implementations. Then, using measurements of the DNS root servers from different locations, we investigate how server selection algorithms affect the performance perceived by users, and load-sharing of the servers.

Our results show that proper use of the server selection algorithms is essential to the performance and the stability of Internet services.

2 Related Work

The center placement problem has been a well-known subject of research, and studied both in theory and practical applications [1]. It is a problem to find the optimal placement of a set of centers or the minimum number of centers for given users.

Although placement of servers in the Internet is similar, there are other practical issues such as distance measurement, fluctuations or uncertainty of the environment, and constant growth of the network. Jamin *et al.* propose distant maps which provides a relative distance between end-hosts, and discuss its application for mirror server placement [2, 3]. It is also shown that the closest server selection performs much better than random server selection. Qiu *et al.* evaluates different placement algorithms for web server replicas by means of simulation [4]. The focus of these approaches is to improve the performance for users, and thus, users are assumed to access the closest server. This paper investigates the load distribution of servers as well as the performance, and focuses on the effects of different server selection algorithms.

As for DNS measurement, Brownlee *et al.* passively observed DNS traffic on a university campus and analyze the behavior of the root and gTLD servers [5]. They also analyze DNS traffic at the F root server [6]. Fomenkov *et al.* investigate the connectivity of the DNS root servers to a large number of DNS clients [7]. The measurement is done by co-locating an active measurement tool, called skitter, with six of the root servers. These DNS results do not consider the effects of server selection algorithms on the user side, which motivated our research on DNS. We make use of measurements of the root servers taken by Sekiya *et al.* [8] for our simulation.

3 Server Placement and Server Selection

Server placement is heavily influenced by the server selection mechanism used for the system. For users to receive good performance, it is important to choose a near server since some servers could be very far on the Internet. In addition, to increase availability, a user should be able to switch to an alternative server if needed. If the nearest server fails, or its performance degrades, a user needs to choose another server to continuously receive the service in good quality.

On the other hand, a service provider should arrange servers so as to provide good system-wide performance. However, load-sharing among the servers is sometimes more important than the performance from the administrator's point of view. It is often needed to place an additional server to reduce the load of heavily-loaded servers. The location of the new server should be chosen carefully in order to distribute the service load appropriately. Furthermore, it is necessary to consider the effects to the performance perceived by users as well. That is, the performance should not be sacrificed too much by load distribution.

In this paper, distance or response time is used as a metric for server selection but there are other metrics such as throughput. Our principle of balancing performance and load distribution in an adaptive way also applies to other metrics.

3.1 Server Selection Algorithms

When a set of servers for a certain service are available, a user selects one of the servers. There are different mechanisms to select a server to use. We introduce three representative server selection algorithms and illustrate the differences.

1. best-server algorithm
2. uniform algorithm
3. reciprocal algorithm

The best-server algorithm measures the conditions of the servers and chooses one as the best server. The metric can be round-trip time, the number of hops, or other kind of network distance. The best server can be chosen from these metrics. This algorithm is optimal in performance but hard to control load-sharing as described later.

The uniform algorithm selects all the servers uniformly. This algorithm can be realized by round-robin or random selection, and does not use any metrics. It is easy to implement this algorithm so that it has been used widely by many network applications. This algorithm is good for load-sharing but poor in performance, especially if a bottleneck server exists.

The reciprocal algorithm selects a server with a probability reciprocal to some metrics. Unlike the best-server algorithm, not only the best server but also other servers are used. The access probability to each server is a function of some metrics. For example, if a distance is used as a metric, a near server is used more frequently than a far server, and two servers located at the same distance are used equally. This algorithm is adaptive in the face of fluctuating server conditions since selection is dynamically determined by the function.

3.2 Algorithm Evaluation

To illustrate the differences of the three algorithms, simple synthetic network configurations are used. In the following examples, the distance between a user and a server is used as the metric for server selection. That is, when user i accesses server j and the distance between i and j is d_{ij}, the access cost is defined as d_{ij}. When a set of servers S is given and the number of the servers is m, The cost function of user i, $c(i)$, for the best-server algorithm is

$$c(i) = \min_{j \in S} d_{ij}$$

For the uniform algorithm, $c(i)$ is simply the average of the distances.

$$c(i) = \frac{1}{m} \sum_{j=1}^{m} d_{ij}$$

For the reciprocal algorithm, the probability of using a server is reciprocal to the distance. Therefore, the probability of user i using server j is

$$p_{ij} = \frac{1}{d_{ij} \sum_{j \in S} \frac{1}{d_{ij}}}$$

The cost function of the reciprocal algorithm is

$$c(i) = \sum_{j \in S} d_{ij} p_{ij} = \frac{m}{\sum_{j \in S} \frac{1}{d_{ij}}}$$

Optimal Placement

The optimal placement is to minimize the total cost. That is, for given set of users U, place a set of servers S so as to

$$minimize \sum_{i \in U} c(i)$$

To illustrate the differences of the algorithms, we use a simple configuration as shown in Figure 1. Assume 16 users are placed at each vertex of a 4×4 square mesh. We consider the optimal arrangement of 4 identical servers for each algorithm.

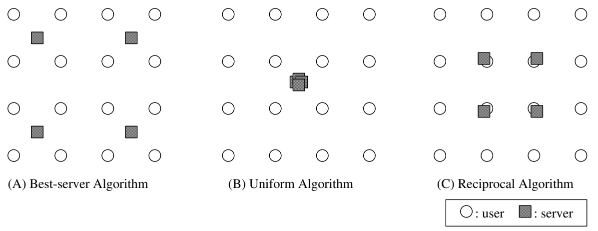

Figure 1: optimal placement of 4 servers for 16 users with different selection algorithms

Figure 1 (A) shows the optimal placement for the best-server algorithm. We need to minimize the average distance from each user to the nearest server. This is known as the k-median problem [1] and a generic solution is NP-hard. However, it is easy to solve in this configuration by dividing the users into 4 clusters and placing a server at the center of each cluster.

Figure 1 (B) shows the optimal placement for the uniform algorithm. We need to minimize the average distance from each user to all servers. In this example, it is optimal to place the 4 servers altogether at the center of the users.

Figure 1 (C) shows the optimal placement for the reciprocal algorithm. When server selection is performed with the reciprocal algorithm, a user uses not only the nearest server but also others with probability reciprocal to the distance between the user and the server.

As easily seen from the figures, the best-server algorithm shows the best performance and the performance of the uniform algorithm is the worst. All servers have the same share of the load for all the algorithms.

Adding a Server for Load-Sharing

It is often needed to place an additional server to reduce the service load of heavily-loaded servers. We consider the effects of an additional server to the system-wide performance.

The network configuration in Figure 2 is used here. 16 users and 4 servers are used as in the previous case but, this time, the locations of the users and the servers are the same for all the algorithms.

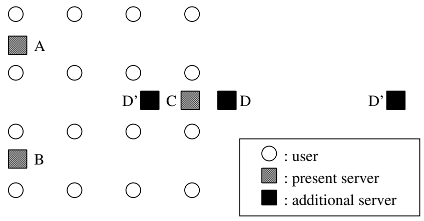

Figure 2: 3 existing servers, a new server at D, at D’ or at D”

The effects of an additional server, D, D' or D'' in the figure, are observed. D and D'' are close to one of the existing servers, C, D' is far from the existing servers and the users. Table 3.2 shows the changes in the load distribution and the average response time by the additional server. The distance of two points is computed as the Euclid distance, and the cost of response time between adjacent vertices is normalized to 1.

Table 1: server load and response time by an additional server

Algorithm		Load Distribution A	B	C	D(’)	Response Time
Best-server		25%	25%	50%	-	1.020
	+D	25%	25%	50%	0%	1.020
	+D’	25%	25%	50%	0%	1.012
	+D”	25%	25%	25%	25%	0.941
Uniform		33%	33%	33%	-	2.116
	+D	25%	25%	25%	25%	2.174
	+D’	25%	25%	25%	25%	2.748
	+D”	25%	25%	25%	25%	2.016
Reciprocal		32%	32%	36%	-	1.614
	+D	26%	26%	27%	21%	1.656
	+D’	29%	29%	32%	10%	1.890
	+D’‘	24%	24%	25%	27%	1.538

In the case of the best-server algorithm, adding D or D' has no effect. The right half of the users use server C and the left half of the users are divided into server A and server B. Both D and D' are behind server C for all the users so that they are not used at all. However, when D'' is added, the load of C is divided between C and D''. This example illustrates a difficulty in controlling load-sharing with the best-server algorithm. Even if a new server is added to the existing overloaded server as is the case for D, it may not help at all. The opposite case is also possible; if a new server is placed just in front of the existing server, all the load could be shifted from the existing server to the new server.

On the other hand, the load is assigned equally to all the servers with the uniform algorithm. Regardless of the position of the new server, the load of each server is reduced from 33% to 25% by adding a server. The response time is, however, affected by the position of the new server. Before adding the new server, the response time of the unform algorithm is already the worst in the 3 algorithms. It degrades slightly by adding D, and degrades severely by adding D' since the users access D' equally. This illustrates a difficulty to control the system-wide performance with the uniform algorithm

In the case of the reciprocal algorithm, D contributes to load-sharing and the response time degrades slightly. The impact of D' is small since its distance is large for all the users. The users still access D' and the performance drops accordingly.

We can infer dynamic condition changes using Table 3.2. If the connectivity to D fluctuates, D is perceived as being at D' or D''. Further, if it fails, only the other 3 servers remain. It is easy to see that the load distribution of the best-server algorithm is heavily affected as the position of D fluctuates. The performance of the uniform algorithm also fluctuates as the position of D fluctuates. On the other hand, the reciprocal algorithm can adapt in both performance and load distribution as D fluctuates.

Although a simple configuration of 16 users is used in this example, it is obvious that our observation also applies to more complex configurations. The observed problems are inherent in the algorithms, and it simply becomes harder to predict the effects as the number of users increases and the user distribution becomes unbalanced.

3.3 Practical Issues

Distance Measurement

So far, we used the distance between a user and a server as our metric. In a real network, however, it is difficult to define the distance. Moreover, a user can obtain limited information about the network and the servers. How to measure the distance on the Internet is still under active research [2, 3].

In practice, the response time from a server is often used instead of the distance to the server. The response time measured by a user includes the network delay, the server processing time and the local processing time.

Since the response time fluctuates, the average response time in the recent past is used to predict the response time in the near feature. Depending on applications, the variance of the response time is used as well.

The network and server conditions can change in a short time. A server selection mechanism should be able to adapt to changes of the situation. How quickly a mechanism adapts to a change depends on how often it measures the condition. The more frequently it measures the condition, the quicker it adapts.

When the response time is used to measure the condition, a user needs to send a request to obtain the response time. The best-server algorithm is not suitable for detecting condition changes since it does not access the servers other than the best server and does not update the response time of the other servers. The reciprocal algorithm can detect condition changes of a near server better than a far server since the near server is accessed more than the far server. This is another advantage of the reciprocal algorithm.

Operational Restrictions

It is often not possible to realize ideal server placement in a real environment due to operational restrictions. There are a limited number of places where servers can be accommodated. Once a server is installed and a service is started, it is not easy to change the configuration even when a need for rearrangement arises. On the other hand, the service may need to stop for hardware or software maintenance. Other unexpected problems could arise such as failures of network or facilities.

As the scale of a service increases, the service becomes more difficult to manage as planned. Therefore, server placement and server selection should be designed to be flexible and fail-safe. To this end, adaptivity is an important property for large-scale services, especially from the operational point of view.

3.4 Summary

In server placement planning, it is important to take server selection mechanisms into consideration. We have illustrated the behavior of three types of server selection algorithms.

The behavior of the best-server algorithm is intuitive and the best performance can be achieved. However, its load distribution is sensitive to the server placement, and hard to control in a real network. A slight change of the environment could lead to an unexpectedly-large shift in load distribution.

The uniform algorithm provides fair load distribution but it is hard to improve the performance. Because each server has the same share, the system performance is dominated by the bottleneck server. In global Internet services, it is usually much more difficult to control the bottleneck server than the best server.

The performance of the reciprocal algorithm is not as good as the best-server algorithm but it is much easier to control load-sharing by server placement. The algorithm is adaptive to condition changes, which is important to make a service robust on the Internet.

For large-scale Internet services, the following properties are needed for a server selection algorithm.

1. An algorithm prefers servers with better performance. It is not only for performance but also allows server placement to control load distribution.

2. An algorithm sends equal load to 2 servers if their performance is equal. This prevents oscillations between 2 servers.

3. An algorithm adapts to condition changes.

The reciprocal algorithm satisfies these properties.

4 A Case Study: DNS

4.1 Domain Name System

DNS translates host names to IP addresses. DNS is a distributed database in which domain names are maintained in a hierarchical tree structure. A domain in the domain name space may be divided into subdomains, and the administration of a subdomain may be delegated. A zone is an administrative unit of the domain name space in which a set of name servers are authoritative for the domain as well as responsible for providing referrals of its delegated subdomains. When a name server at the client side is asked to resolve a name, it traverses the name hierarchy and sends queries recursively to an authoritative server of each zone within the specified name. DNS also uses caching extensively to reduce repetitive queries for the same zone.

A zone can have multiple authoritative servers for better performance and robustness. When there are multiple name servers authoritative for a zone, a recursive server picks up one to send a query. How to select a server is implementation dependent. The DNS specifications [9, 10] suggest to sort the server list by statistics such as previous response times and batting average.

4.2 DNS Implementations

There are several DNS implementations which employ different server selection mechanisms. These mechanisms can be categorized into the three algorithms described in the previous section.

BIND-8

The Berkeley Internet Name Domain system (BIND) [11] is the most widely used implementation of DNS. The server selection algorithm of *BIND-8*, version 8 of *BIND*, can be viewed as a variant of the reciprocal algorithm in the sense that the access probability is a function of server response time. The older versions of *BIND* also have the same algorithm.

BIND-8 maintains the list of name servers for a zone. When *BIND-8* finds multiple name servers to resolve a request, it sorts the servers by the smoothed response time, and tries the servers in this order[1]. The smoothed response time is the average response time of this server in the recent past.

The smoothed response time is maintained as follows. When a response comes back, the smoothed average response time, $srtt$, is computed using the exponentially-weighted moving average:

$$srtt = \alpha \cdot srtt + (1 - \alpha) \cdot rtt \tag{1}$$

Then, the entries of the unused servers in the list are decayed by

$$srtt = \gamma \cdot srtt \tag{2}$$

By slowly reducing $srtt$ of the unused servers, they are eventually tried again[2]. *BIND-8* uses $(\alpha = 0.7)$ and $(\gamma = 0.98)$.

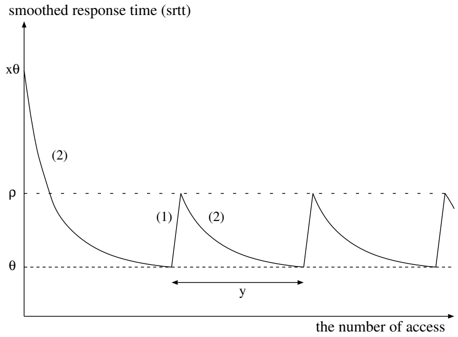

Figure 3: a model of smoothed response time in BIND-8

Figure 3 illustrates the effect of the algorithm. For simplicity, we assume a server has a constant response time and there is a constant threshold to select a server. When a server is accessed for the first time, the response time is recorded. While this server is not used, $srtt$ is slowly decayed by Equation 2 every time other servers for the same name are referenced. Eventually, $srtt$ hits the threshold θ, and this server is used again. This time, $srtt$ is increased by Equation 1. Then, this server is not used until $srtt$ hits the threshold again.

In the steady state, $srtt$ follows a sawtooth track, and the server is selected once in y access. In other words, the expected share of the server is $1/y$.

Let the constant response time be x times larger than θ. That is, $rtt = x \cdot \theta$. Let ρ be the peak value of the sawtooth track. When the server is accessed, $(srtt = \theta)$. From Equation 1,

$$\rho = \theta(\alpha + x(1 - \alpha)) \tag{3}$$

The server is not used for the next $(y - 1)$ times, and becomes θ again. From Equation 2

$$\gamma^{(y-1)} \cdot \rho = \theta \tag{4}$$

[1] *BIND* maintains separate entries for cached data and for data read from the file. We assume the cached entry and its TTL is long enough

[2] *BIND* also penalizes those who had earlier chances but have not responded. This algorithm is omitted in this analysis for simplicity.

From Equation 3 and 4, we can eliminate θ and ρ.

$$\gamma^{(y-1)} = \frac{1}{\alpha + (1-\alpha)x}$$
$$y - 1 = \log_\gamma \frac{1}{\alpha + (1-\alpha)x}$$
$$y = 1 - \frac{\log(\alpha + (1-\alpha)x)}{\log\gamma} \quad (5)$$

By applying $(\alpha = 0.7)$ and $(\gamma = 0.98)$

$$y = 1 - \frac{\log(0.7 + 0.3x)}{\log 0.98} \quad (6)$$

Figure 4 plots how selection cycle y changes with varying response time factor x. We also plot $(y = x)$ and $(y = x^2)$ for comparison. $(y = x)$ corresponds to the reciprocal algorithm.

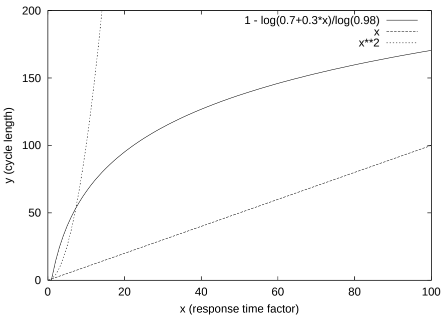

Figure 4: the server selection cycle of BIND-8 with varying response time

The *BIND-8* algorithm employs a concave function instead of a linear function. Intuitively, it magnifies the difference when a server is close, and minifies the difference when the server is far away. This prevents the access probability of a far server from being too small. It allows a user to keep track of all the servers, and is suitable when the server set is relatively small as is the case for DNS. On the other hand, if the server set is large, a convex function is suitable to ignore far servers. It is also an effective way to reduce the effect of poorly-performing servers.

A concave function of *BIND-8* also has a bias to select the nearest server more than a linear function. The *BIND-8* algorithm seems to be a little aggressive to segregate near servers. When the distance ratio of two servers is $1 : 2$, the ratio of their access probability is $1 : 0.07$. When the distance ratio is $1 : 5$, the access ratio is $1 : 0.025$.

One way to distribute the load more to other near servers is to use larger α, although α is the weight for moving average and larger α means slower convergence. If α is set to 0.9 instead of 0.7, the access ratio becomes $1 : 0.18$ for distance ratio $1 : 2$, and $1 : 0.056$ for distance ratio $1 : 5$.

On a side note, the *BIND-8* implementation directly applies Equation 2 to $srtt$ for aging, and does not keep track of the real measured response time. It might be useful to have a separate variable for the purpose of server selection since the response time can be used for other purposes such as identifying a malfunctioning server.

BIND-9

The algorithm of *BIND-9* is a variant of the best-server algorithm. *BIND-9* implements Equation 1 but not Equation 2 [3]. The $srtt$ parameter for a name server is initialized to a small random value so that all name servers are accessed at least once. However, a recursive server will refer to only the best performing server, once $srtt$ of the other servers are recorded.

This could be a problem in some situations. The algorithm is adaptive only when the best performing server slows down. It is unable to detect a situation where the performance of another server improves. It is possible that a recursive server switches to a non-optimal server under a short outage but never goes back to the best server thereafter.

DJBDNS and Windows Internet Server

The algorithm of *djbdns* [12] selects one in the server list randomly, and can be categorized into the uniform algorithm.

Also, our experiments indicate that the name server implementation of Microsoft Windows 2000 Internet Server is in this category, although we could not find any reference to confirm our experiment result.

4.3 Evaluation by Root Server Measurement

In this section, we apply different server-selection algorithms to a data set measured on the Internet in order to observe their effects to the real environment. From the response time of a set of servers, we can derive the expected load distribution and the expected average response time for each client with the different algorithms.

Note that the simulations are used merely to observe the effects of different server selection algorithms in more realistic situations. Because of the limitations of the data set used for simulation, the results are not intended to show specific DNS issues on specific locations or their response time.

[3]later, *BIND-9* was changed to implement the same algorithm as *BIND-8* since version 9.2.2.

Table 2: the median response time (msec) of the root servers from different locations

Measurement	Root Servers												
Point	A	B	C	D	E	F	G	H	I	J	K	L	M
US(1)	88	22	520	75	16	24	385	80	203	89	163	38	134
US(2)	79	21	545	67	2	2	374	72	183	79	152	24	123
US(3)	72	135	521	315	178	111	499	316	437	71	236	140	105
US(4)	2	70	430	6	64	76	315	4	116	3	79	75	192
US(5)	4	67	477	1	70	82	275	5	135	2	89	92	189
US(6)	22	76	449	9	70	82	200	15	131	23	93	94	192
CA*	140	200	570	140	371	181	461	160	220	120	191	200	330
MX*	110	91	101	100	131	100	290	90	200	81	170	100	211
UK	190	179	542	105	170	170	310	114	57	110	72	184	254
FR	116	188	540	108	193	148	397	152	32	112	32	179	251
CH	96	178	514	112	163	158	258	115	58	96	27	199	300
IT*	200	251	630	150	270	220	347	160	100	170	70	220	331
PL*	170	220	660	140	361	200	361	150	90	150	80	230	356
UA*	180	501	620	440	270	250	620	451	350	160	350	500	590
CN(1)*	280	401	930	220	551	400	591	470	371	480	351	151	421
CN(2)*	750	670	1190	720	250	360	910	720	820	710	521	660	540
KR*	310	220	980	291	281	201	671	290	400	291	360	231	220
JP	178	140	614	169	102	100	430	170	270	170	230	137	1
NZ	209	137	648	202	146	135	434	206	307	201	270	150	160
AU*	360	270	800	381	390	250	705	320	480	321	440	250	200
ZA*	348	388	808	308	489	378	498	298	338	308	378	389	508
KE*	329	359	489	250	-	340	480	369	399	350	360	330	490
DZ*	210	280	630	181	250	250	351	180	140	180	100	280	350
BR(1)*	140	161	541	111	161	151	101	101	211	101	181	181	251
BR(2)	140	198	555	149	190	194	327	125	248	141	216	196	303
AR	171	203	613	163	222	220	364	167	270	163	243	203	322
CL*	140	220	571	140	210	180	481	140	250	140	220	181	310

Measurement Data

The data set includes the response time of the DNS root name servers measured from different locations around the world. Currently, there are 13 root name servers named from 'A' to 'M'; 6 in the East Coast, 4 in the West Coast, 2 in Europe, and 1 in Japan.

The measurements were collected from 27 locations around the world in May and June, 2002 [8]. It uses an active measurement tool which sends DNS queries to the root servers and measures the response time. For locations where setting up the tool is difficult, dialup from Japan is performed, and the results are compensated for the delay caused by the dialup access.

The median of the measured response time for each root server is shown in Table 2. The response time differs in orders of magnitude since the root servers are distributed around the world. The measurement points are shown by their country codes. The dialup points have '$*$' after the country code.

Although the measurement points are classified by their country codes, the data does not necessarily reflect a typical view from the country because the measurement points are selected based on ease of access and have different topology and an access line type to the Internet. The time of measurement also varies for different locations. Nonetheless, it shows a real view of a set of servers observed from different locations around the world.

We do not use information of response loss in our simulation since there is no standard or simple way to reflect the loss rate. However, response-loss is an important factor to select a server. Even if the response time is small, the loss rate could be high for a server. Obviously, such a server is not good. To take *BIND-8* as an example, *BIND-8* penalizes $srtt$ by 20% when it does not receive a response.

We also do not consider the effects of caching in our simulation. Caching does not affect the load distribution but significantly reduces the perceived response time and the traffic.

4.4 Simulation Results

The different server selection algorithms are applied to the measurement data in order to observe the effects of the algorithms. In addition to the 3 basic algorithms described in the previous section, 2 variants of the reciprocal algorithms are used; one uses $1/d^2$ instead of $1/d$ and the other uses the *BIND-8* algorithm. The simulation results are shown in Table 3 through 7.

For each algorithm, the expected load distribution and the expected average response time are computed from the measured response time of the root servers.

Regarding the performance, The best-server algorithm is optimal in this simulation. The performance of the uniform algorithm is poor due to large variations in the response time of the servers. For a global Internet service, it is unavoidable that some servers are located on the other side of the planet, which is an adverse condition for the uniform algorithm.

As for the load distribution, Table 3 shows that B, C and G root servers are not used by the best-server algo-

Table 3: simulation results of the best-server algorithm

Measurement						Load Distribution (%)								Response
Point	A	B	C	D	E	F	G	H	I	J	K	L	M	Time (msec)
US(1)	0	0	0	0	100	0	0	0	0	0	0	0	0	16.0
US(2)	0	0	0	0	100	0	0	0	0	0	0	0	0	2.0
US(3)	0	0	0	0	0	0	0	0	0	100	0	0	0	71.0
US(4)	100	0	0	0	0	0	0	0	0	0	0	0	0	2.0
US(5)	0	0	0	100	0	0	0	0	0	0	0	0	0	1.0
US(6)	0	0	0	100	0	0	0	0	0	0	0	0	0	9.0
CA*	0	0	0	0	0	0	0	0	0	100	0	0	0	120.0
MX*	0	0	0	0	0	0	0	0	0	100	0	0	0	81.0
UK	0	0	0	0	0	0	0	0	0	100	0	0	0	57.0
FR	0	0	0	0	0	0	0	0	100	0	0	0	0	32.0
CH	0	0	0	0	0	0	0	0	0	0	100	0	0	27.0
IT*	0	0	0	0	0	0	0	0	0	0	100	0	0	70.0
PL*	0	0	0	0	0	0	0	0	0	0	100	0	0	80.0
UA*	0	0	0	0	0	0	0	0	0	100	0	0	0	160.0
CN(1)*	0	0	0	0	0	0	0	0	0	0	0	100	0	151.0
CN(2)*	0	0	0	0	100	0	0	0	0	0	0	0	0	250.0
KR*	0	0	0	0	0	100	0	0	0	0	0	0	0	201.0
JP	0	0	0	0	0	0	0	0	0	0	0	0	100	1.0
NZ	0	0	0	0	0	100	0	0	0	0	0	0	0	135.0
AU*	0	0	0	0	0	0	0	0	0	0	0	0	100	200.0
ZA*	0	0	0	0	0	0	0	100	0	0	0	0	0	298.0
KE*	0	0	0	100	0	0	0	0	0	0	0	0	0	250.0
DZ*	0	0	0	0	0	0	0	0	0	0	100	0	0	100.0
BR(1)*	0	0	0	0	0	0	0	100	0	0	0	0	0	101.0
BR(2)	0	0	0	0	0	0	0	100	0	0	0	0	0	125.0
AR	0	0	0	0	0	0	0	0	0	100	0	0	0	163.0
CL*	0	0	0	0	0	0	0	0	0	100	0	0	0	140.0

Table 4: simulation results of the uniform algorithm

Measurement						Load Distribution (%)								Response
Point	A	B	C	D	E	F	G	H	I	J	K	L	M	Time (msec)
US(1)	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	141.3
US(2)	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	132.5
US(3)	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	241.2
US(4)	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	110.2
US(5)	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	114.5
US(6)	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	112.0
CA*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	252.6
MX*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	136.5
UK	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	189.0
FR	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	188.3
CH	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	174.9
IT*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	239.8
PL*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	243.7
UA*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	406.3
CN(1)*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	432.1
CN(2)*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	678.5
KR*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	365.1
JP	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	208.5
NZ	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	246.5
AU*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	397.5
ZA*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	418.2
KE*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	378.75
DZ*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	260.2
BR(1)*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	184.0
BR(2)	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	229.4
AR	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	255.7
CL*	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	7.7	244.9

rithm. This is due to the fact that the measurement points are very limited in our data set, and it is unlikely that these servers are not used in the real environment even if all the users use the best-server algorithm. Still, the algorithm is sensitive to the server locations, and the results indicate difficulties in arranging server locations.

When the 3 variants of the reciprocal algorithms are compared, there is a trade-off between performance and load distribution. Better load distribution is obtained in exchange for poorer performance. As explained in section 4.2, the *BIND-8* has a bias towards the best-server but still access far servers more than the other two. The $1/d^2$ algorithm has a strong bias against far servers.

We believe that the *BIND-8* algorithm is fairly reason-

Table 5: simulation results of the reciprocal algorithm ($1/d$)

Measurement Point	Load Distribution (%) A	B	C	D	E	F	G	H	I	J	K	L	M	Response Time (msec)
US(1)	4.6	18.4	0.8	5.4	25.3	16.8	1.0	5.1	2.0	4.5	2.5	10.6	3.0	52.5
US(2)	1.1	4.1	0.2	1.3	42.8	42.8	0.2	1.2	0.5	1.1	0.6	3.6	0.7	11.1
US(3)	16.6	8.9	2.3	3.8	6.7	10.8	2.4	3.8	2.7	16.9	5.1	8.6	11.4	155.8
US(4)	37.4	1.1	0.2	12.5	1.2	1.0	0.2	18.7	0.6	24.9	0.9	1.0	0.4	9.7
US(5)	12.3	0.7	0.1	49.2	0.7	0.6	0.2	9.8	0.4	24.6	0.6	0.5	0.3	6.4
US(6)	13.1	3.8	0.6	31.9	4.1	3.5	1.4	19.2	2.2	12.5	3.1	3.1	1.5	37.4
CA*	11.2	7.8	2.7	11.2	4.2	8.7	3.4	9.8	7.1	13.1	8.2	7.8	4.7	203.7
MX*	8.3	10.0	9.0	9.1	6.9	9.1	3.1	10.1	4.5	11.2	5.3	9.1	4.3	356.7
UK	5.5	5.8	1.9	9.9	6.1	6.1	3.4	9.1	18.3	9.5	14.5	5.7	4.1	135.6
FR	6.8	4.2	1.5	7.3	4.1	5.3	2.0	5.2	24.6	7.0	24.6	4.4	3.1	102.3
CH	8.3	4.5	1.6	7.1	4.9	5.1	3.1	7.0	13.8	8.3	29.6	4.0	2.7	104.0
IT*	6.9	5.5	2.2	9.2	5.1	6.3	4.0	8.6	13.8	8.1	19.7	6.3	4.2	179.7
PL*	8.0	6.2	2.1	9.7	3.8	6.8	3.8	9.0	15.1	9.0	17.0	5.9	3.8	176.3
UA*	14.4	5.2	4.2	5.9	9.6	10.4	4.2	5.7	7.4	16.2	7.4	5.2	4.4	336.5
CN(1)*	9,8	6.8	3.0	12.5	5.0	6.9	4.6	5.8	7.4	5.7	7.8	18.2	6.5	356.7
CN(2)*	6.0	6.7	3.8	6.2	18,0	12.5	4.9	6.2	5.5	6.3	8.6	6.8	8.3	584.7
KR*	7.5	10.5	2.4	7.9	8.2	11.5	3.4	8.0	5.8	7.9	6.4	10.0	10.5	300.3
JP	0.5	0.7	0.2	0.6	0.9	0.9	0.2	0.5	0.3	0.5	0.4	0.7	93.5	12.2
NZ	7.4	11.3	2.4	7.6	10.6	11.4	3.6	7.5	5.0	7.7	5.7	10.3	9.6	200.5
AU*	7.3	9.8	3.3	6.9	6.8	10.5	3.7	8.2	5.5	8.2	6.0	10.5	13.2	342.7
ZA*	8.6	7.7	3.7	9.7	6.1	7.9	6.0	10.1	8.9	9.7	7.9	7.7	5.9	389.5
KE*	9.3	8.5	6.2	12.2	0.0	9.0	6.3	8.3	7.6	8.7	8.5	9.2	6.2	365.7
DZ*	7.8	5.9	2.6	9.1	6.6	6.6	4.7	9.1	11.7	9.1	16.4	5.9	4.7	213.5
BR(1)*	8.3	7.2	2.1	10.4	7.2	7.7	11.4	11.4	5.5	11.4	6.4	6.4	4.6	213.5
BR(2)	10.8	7.6	2.7	10.1	7.9	7.8	4.6	12.1	6.1	10.7	7.0	7.7	5.0	196.0
AR	10.0	8.4	2.8	10.5	7.7	7.8	4.7	10.3	6.4	10.5	7.1	8.4	5.3	222.9
CL*	11.1	7.0	2.7	11.1	7.4	8.6	3.2	11.1	6.2	11.1	7.0	8.6	5.0	201.4

Table 6: simulation results of the reciprocal algorithm ($1/d^2$)

Measurement Point	Load Distribution (%) A	B	C	D	E	F	G	H	I	J	K	L	M	Response Time (msec)
US(1)	1.4	22.7	0.0	1.9	42.8	19.0	0.1	1.7	0.3	1.4	0.4	7.6	0.6	27.1
US(2)	0.0	0.4	0.0	0.0	49.5	49.5	0.0	0.0	0.0	0.0	0.0	0.3	0.0	2.3
US(3)	25.7	7.3	0.5	1.3	4.2	10.8	0.5	1.3	0.7	26.4	2.4	6.8	12.1	111.1
US(4)	55.3	0.0	0.0	6.1	0.1	0.0	0.0	13.8	0.0	24.6	0.0	0.0	0.0	3.0
US(5)	4.6	0.0	0.0	73.9	0.0	0.0	0.0	3.0	0.0	18.5	0.0	0.0	0.0	1.5
US(6)	9.6	0.8	0.0	57.1	0.9	0.7	0.1	20.6	0.3	8.7	0.5	0.5	0.1	16.1
CA*	14.0	6.9	0.8	14.0	2.0	8.4	1.3	10.7	5.7	19.1	7.5	6.9	2.5	175.6
MX*	8.0	11.7	9.5	9.7	5.7	9.7	1.2	12.0	2.4	14.8	3.4	9.7	2.2	107.0
UK	3.0	3.3	0.4	9.7	3.7	3.7	1.1	8.2	32.8	8.8	20.6	3.1	1.7	102.2
FR	3.1	1.2	0.1	3.6	1.1	1.9	0.3	1.8	40.8	3.3	40.8	1.3	0.7	53.1
CH	4.9	1.4	0.2	3.6	1.7	1.8	0.7	3.4	13.5	4.9	62.2	1.1	0.5	56.7
IT*	4.6	3.0	0.5	8.3	2.5	3.8	1.5	7.3	18.6	6.4	37.9	3.8	1.7	134.5
PL*	6.4	3.8	0.4	9.4	1.4	4.6	1.4	8.2	22.7	8.2	28.7	3.5	1.4	135.5
UA*	21.7	2.8	1.8	3.6	9.7	11.3	1.8	3.5	5.7	27.5	5.7	2.8	2.0	271.8
CN(1)*	10.0	4.9	0.9	16.3	2.6	4.9	2.3	3.6	5.7	3.4	6.4	34.5	4.4	287.1
CN(2)*	3.8	4.8	1.5	4.2	34.5	16.6	2.6	4.2	3.2	4.3	7.9	5.0	7.4	479.5
KR*	6.5	12.9	0.6	7.4	7.9	15.4	1.4	7.4	3.9	7.4	4.8	11.7	12.9	269.5
JP	0	0	0	0	0	0	0	0	0	0	0	0	100	1.1
NZ	6.2	14.5	0.7	6.7	12.8	15.0	1.4	6.4	2.9	6.8	3.7	12.1	10.7	177.0
AU*	6.2	11.0	1.3	5.5	5.3	12.9	1.6	7.8	3.5	7.8	4.2	12.9	20.1	304.8
ZA*	9.2	7.4	1.7	11.7	4.6	7.8	4.5	12.5	9.7	11.7	7.8	7.3	4.3	370.0
KE*	9.9	8.3	4.5	17.2	0.0	9.3	4.7	7.9	6.8	8.8	8.3	9.9	4.5	352.8
DZ*	6.6	3.7	0.7	8.9	4.7	4.7	2.4	9.0	14.9	9.0	29.2	3.7	2.4	178.2
BR(1)*	7.8	5.9	0.5	12.5	5.9	6.7	15.1	15.1	3.5	15.1	4.7	4.7	2.4	178.2
BR(2)	13.5	6.8	0.9	11.9	7.3	7.0	2.5	17.0	4.3	13.3	5.7	6.9	2.9	175.8
AR	12.0	8.5	0.9	13.2	7.1	7.2	2.6	12.6	4.8	13.2	5.9	8.5	3.4	204.4
CL*	14.0	5.7	0.8	14.0	6.2	8.5	1.2	14.0	4.4	14.0	5.7	8.4	2.9	177.7
DZ*	6.6	3.7	0.7	8.9	4.7	4.7	2.4	9.0	14.9	9.0	29.2	3.7	2.4	178.2

able in terms of both performance and load distribution. The best-server algorithm in old *BIND-9* is good for performance when the system environment is stable. However, it is not flexible in the face of a condition change and the behavior is not predictable when uncertainty exists in the environment. The uniform algorithm in *djbdns* or *Microsoft* server is suitable if the performance does not matter or if all servers show similar performance to all users.

Currently, *BIND-8* has the majority of the installed base of name servers but the share of *BIND-9* and Windows Internet Server is increasing. If the best-server al-

Table 7: simulation results of the BIND-8 algorithm

Measurement	Load Distribution (%)													Response
Point	A	B	C	D	E	F	G	H	I	J	K	L	M	Time (msec)
US(1)	2.2	13.6	0.8	2.5	55.1	11.0	0.9	2.4	1.3	2.2	1.4	5.1	1.6	38.9
US(2)	0.8	1.4	0.4	0.8	45.7	45.7	0.5	0.8	0.6	0.8	0.6	1.3	0.7	11.7
US(3)	27.0	5.9	1.7	2.4	4.1	8.3	1.7	2.4	1.9	27.0	3.1	5.6	9.0	127.9
US(4)	71.3	0.8	0.6	3.9	0.8	0.8	0.5	6.8	0.7	11.8	0.8	0.8	0.6	10.7
US(5)	3.0	0.7	0.4	82.8	0.6	0.6	0.4	2.4	0.5	6.9	0.6	0.6	0.5	8.4
US(6)	5.0	1.6	0.7	69.2	1.7	1.6	1.0	9.4	1.2	4.7	1.4	1.4	1.0	24.8
CA*	12.9	6.5	2.1	12.9	3.1	7.6	2.5	9.6	5.6	20.3	6.9	6.4	3.4	187.6
MX*	7.6	11.2	9.0	9.0	5.8	9.0	2.5	11.2	3.5	14.8	4.2	9.0	3.3	112.4
UK	3.3	3.5	1.5	7.0	3.7	3.7	2.1	6.2	41.3	6.5	15.4	3.4	2.5	108.4
FR	3.0	2.0	1.1	3.2	2.0	2.4	1.2	2.4	38.0	3.1	38.0	2.1	1.6	67.9
CH	3.3	1.9	1.0	2.8	2.1	2.1	1.5	2.8	6.1	3.3	69.9	1.8	1.4	62.2
IT*	4.0	3.1	1.5	5.8	2.9	3.6	2.4	5.2	11.9	4.9	48.7	3.6	2.4	136.6
PL*	5.4	3.9	1.6	7.2	2.4	4.3	2.4	6.5	20.0	6.5	33.7	3.7	2.4	145.9
UA*	19.3	3.4	2.8	3.8	7.6	8.6	2.7	3.8	5.1	31.6	5.1	3.4	2.9	281.5
CN(1)*	7.0	4.3	2.0	11.1	3.0	4.3	2.9	3.6	4.7	3.5	5.1	44.6	4.0	283.5
CN(2)*	3.8	4.2	2.4	3.9	44.4	11.3	3.0	3.9	3.4	3.9	5.8	4.2	5.6	465.2
KR*	6.4	12.2	1.9	7.1	7.5	15.3	2.6	7.1	4.5	7.1	5.1	10.9	12.2	285.7
JP	0.5	0.5	0.4	0.5	0.6	0.6	0.4	0.5	0.4	0.5	0.5	0.5	94.1	13.4
NZ	6.4	14.3	1.9	6.4	12.0	14.3	2.7	6.4	3.8	6.4	4.4	11.4	9.8	189.0
AU*	5.9	9.8	2.5	5.5	5.4	11.4	2.8	7.1	4.1	7.1	4.6	11.4	22.3	316.4
ZA*	8.8	7.2	2.9	11.3	5.0	7.5	5.0	12.0	9.2	11.2	7.5	7.2	5.0	378.4
KE*	9.3	8.0	5.1	18.8	0.0	8.8	5.1	7.7	6.7	8.3	8.0	9.3	5.0	353.4
DZ*	5.7	3.9	1.9	7.0	4.5	4.5	3.1	7.0	11.4	7.0	37.0	3.9	3.1	180.7
BR(1)*	7.2	5.8	1.8	11.7	5.8	6.5	14.5	14.5	4.1	14.5	5.0	5.0	3.4	140.9
BR(2)	12.6	6.6	2.2	11.0	6.8	6.6	3.4	17.3	4.8	12.6	5.7	6.6	3.7	184.1
AR	11.5	8.0	2.3	12.6	7.0	7.0	3.6	12.1	5.2	12.6	6.0	8.0	4.2	213.4
CL*	13.4	5.8	2.1	13.4	6.1	7.8	2.5	13.4	4.8	13.4	5.8	7.8	3.8	189.2

gorithm or the unform algorithm becomes dominant, it could have an impact to the stability of DNS. On the other hand, it could contribute to the stability of the DNS service to improve server selection algorithms in DNS implementations.

It should be noted that DNS is a unique service for its importance as an Internet infrastructure and for its unparalleled scale. In addition, the maximum number of authoritative servers for a zone is currently limited only to 13 to fit a response into a single packet with the minimum size.

Among name servers, the root and top level domain servers have special significance. The entire system of DNS relies on these servers that need to serve the whole Internet. The root servers currently processes about 5,000 queries per second [6].

Therefore, our discussion on DNS and the root servers is not necessarily applied to other services on the Internet. Still, we believe that understanding the issues is essential for possible future services with the same level of scale as DNS.

5 Conclusion

We have identified that a server selection mechanism plays an important role in server placement, and evaluated different server selection algorithms from the operational point of view. In a real environment, simple methods such as the best server selection or the uniform server selection may not work as expected due to uncertainties of the working environment.

For large-scale Internet services, the following properties are needed for a server selection algorithm. (1) An algorithm prefers servers with better performance. It is not only for performance but also allows server placement to control load distribution. (2) An algorithm sends equal load to 2 servers if their performance is equal. This prevents oscillations between 2 servers. (3) An algorithm adapts to condition changes. The reciprocal algorithm or its variant satisfies these properties and is more suitable for Internet services.

Then, we have examined practical issues by looking into the different server selection algorithms of the existing DNS implementations. The effects of the different algorithms are shown by simulation using measurements of the DNS root servers.

The results indicate that it could contribute to the stability of the DNS service to improve server selection algorithms in DNS implementations. As DNS becomes increasingly important as an infrastructure of the Internet, it is time to seriously study server selection algorithms in DNS implementations.

References

[1] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys. “A constant-factor approximation algorithm for the k-median problem”. ACM Symposium on Theory of Computing, 1999.

[2] P. Francis, S. Jamin, V. Pazson, L. Zhang, D. Gryniewicz, and Y. Jin, “An architecture for a

globa internet host distance estimation service," INFOCOM, March 1999.

[3] S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang, "On the Placement of Internet Instrumentation", INFOCOM, March 2000.

[4] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "One The Placement of Web Server Replicas", INFOCOM, 2001.

[5] N. Brownlee, kc Claffy, and E. Nemeth, "DNS Root/gTLD Performance Measurements", USENIX LISA, 2001.

[6] N. Brownlee, kc Claffy, and E. Nemeth, "DNS Measurement at a Root Server", Globecom, 2001.

[7] M. Fomenkov, kc claffy, B. Huffaker, and D. Moore, "Macroscopic Internet Topology and Performance Measurements From the DNS Root Name Servers", USENIX LISA, December 2001.

[8] Y. Sekiya, K. Cho, A. Kato, R. Somegawa, T. Jinmei and J. Murai. "DNS root servers observed from worldwide locations", a draft paper, 2002.

[9] P. Mockapetris. Domain names - concepts and facilities. RFC1034, IETF, November 1987.

[10] P. Mockapetris. Domain names – implementation and specification. RFC1035, IETF, November 1987.

[11] ISC BIND, http://www.isc.org/

[12] DJBDNS, http://www.djbdns.org/