The DNS Singularity Uptime Beyond the Black Hole

Erjie Zhang erjzhang@ucdavis.edu Hezhi Xie hezxie@ucdavis.edu Muhammad Hassnain mhassnain@ucdavis.edu Zeerak Babar zebabar@ucdavis.edu

Goals of the Project

- Develop a DNS system against DoS Attack
 - *Replicate data among different servers*
 - Balance load among different servers for internet queries
- Test our system against simulated attacks



Our System Design

Results

- Successfully built a DNS system with multiple name servers
 - Multiple name servers built on multiple virtual machines
- Successfully replicate data among different servers
 - Primary-secondary zone transfer
 - *Replicate automatically* & *periodically*
- Successfully implement load balancing among different servers
 - Least Connection algorithm
 - Proactive load balancing by configuring proxy response limits and timeout duration
- Successfully test and compare the results with simulated attack
 - High volumes of queries sent at one time
 - 100% response rate comparing to 99.1% response rate of single server

Pictures and/or Diagrams

We simulated an attack with a high volume of queries on our system, and the experiment result shows the robustness of our system in this scenario.

- Scenario 1: Single server;
- Scenario 2: Two servers with load balancing;
- Scenarios 3: Six servers with load balancing.

Scenario	Number of Requests	Number of Answers	Respond Rate
1	50000	49551	99.1%
2	50000	49968	99.9%
3	50000	50000	100.0%

Response Rate Under Different Scenarios

APPENDIX	1. Introduction
	2. Proposed Solutions and Challenges
	3. Our Solution
	4. Implementation
	5. Conclusion







Introduction



What is Domain Name System (DNS)?





What is Denial of Service (DoS) Attack?



Three Main Types of DoS Attack

Flood attack

Flood the DNS server with a large volume of queries

Amplification attack

Flood the DNS server with short requests which require long responses

Reflection attack

Manipulate open DNS servers to overwhelm the target victim



What are Primary Defences against DoS Attack?

General	Filtering				
Disable unused services;Install latest security patches;	 Detect and block malicious traffic; Statistically analyze gueries information 				
 Locate DNS servers separately; … 	such as IP addresses, protocol type, port number,				
Toleran	ce-based				
Bandwidth capacity	Server capacity				
Content Distribution Network Smart DNS resolution	 Utilize multiple DNS servers (Data replication, load balancing, …) 				

Smart DNS resolution

• Enhance the ability of DNS servers



What Are We Going To Do?



* The scope of our project does not include security during the replication process.





Proposed Solutions

and Challenges



Proposed solutions

Data replication

• The basic data replication method for DNS is called Zone Transfer



Step1: The primary server loads zone data from designated zone files;

Step2: The secondary server loads zone data through a zone transfer operation (AXFR or IXFR);

Step3: Upon any modifications in the primary server's zone file, the primary server will notify the secondary server through a process named DNS NOTIFY, prompting it to initiate a zone transfer.



Proposed solutions

Load Balancing

- Anycast DNS
 - DNS servers at diverse locations are assigned the same IP address
 - Redirect DNS queries to the nearest server
 - Need the help of ISP, which is difficult to implement
- Load Balancer
 - DNS servers built on multiple virtual machines and are assigned with different IP addresses
 - A load balancer acts as a gateway for client queries, and distribute traffic to different servers
 - It is feasible for us to implement







Our Solution



Our Solution

- Deploy Virtual Machines (VMs) to create a network of DNS servers, including a primary server and multiple secondary servers;
- Each secondary server maintains identical zone files as the primary server through Zone Transfer;
- Deploy a load balancer across these DNS servers, which acts as a gateway for client queries, distributing traffic across the servers.





15





Implementation



Step 1 Build DNS Servers

- Environment: Ubuntu 22
- DNS Tool: Berkeley Internet Name Domain (BIND 9)

No.	Role	IP
1	Client	192.168.64.100
2	Load balancer	192.168.64.2
3	Primary name server	192.168.64.10
4	Secondary name server 1	192.168.64.11
5	Secondary name server 2	192.168.64.12
6	Secondary name server 3	192.168.64.13
7	Secondary name server 4	192.168.64.14
8	Secondary name server 5	192.168.64.15



Table 1: IP Addresses of VMs

Step 1 Build DNS Servers

- Environment: Ubuntu 22
- DNS Tool: Berkeley Internet Name Domain (BIND 9)

hezhi@client:/etc\$ dig www.example.com					hezhi@client:/etc\$ dig	<pre>hezhi@client:/etc\$ dig www.google.com</pre>				
; <<>> DiG 9.18.12-0ub ;; global options: +cr ;; Got answer: ;; ->>HEADER<<- opcode ;; flags: qr aa rd ra;	buntu0.22. md e: QUERY, ; QUERY: 1	04.3-l status , ANSM	/buntu << :: NOERRO /ER: 1, A	>> www.example.com R, id: 33822 UTHORITY: 0, ADDITI	; <<>> DiG 9.18.12-0ubu ;; global options: +cmd ;; Got answer: ;; ->>HEADER<<- opcode: ;; flags: qr rd ra; QUE	QUERY,	status ANSWER:	ountu <<: : NOERRO 1, AUTH	>> www.google.com R, id: 51874 DRITY: 0, ADDITIONAL: 1	
;; OPT PSEUDOSECTION: ; EDNS: version: 0, f1 ; COOKIE: a44ac98d6c0a ;; QUESTION SECTION: :www.example.com.	lags:; udp ab1a801000	: 1232 000657 TN	2 76bfd81cd A	299ab5b419b29 (good	;; OPT PSEUDOSECTION: ; EDNS: version: 0, fla ; COOKIE: c1e5f270907d2 ;; QUESTION SECTION: ;www.google.com.	gs:; ud 6bf0100	p: 1232	5cdde93f	bb966f4edacbe (good) A	
;; ANSWER SECTION: www.example.com.	259200	IN	A	10.0.0.7	;; ANSWER SECTION: www.google.com.	300	IN	A	142.250.189.228	
;; Query time: 0 msec ;; SERVER: 192.168.64 ;; WHEN: Mon Dec 11 07	.10#53(192 7:52:56 UT	.168.6 C 2023	54.10) (U	DP)	;; Query time: 212 msec ;; SERVER: 192.168.64.1 ;; WHEN: Mon Dec 11 08: ;; MSG SIZE rcvd: 87	.0#53(19 52:46 U	2.168.64 TC 2023	4.10) (U	DP)	

18

UCDAVIS

Step 2 Data Replication

• Method: Primary-secondary zone transfer

Dec 10 10:41:44 ns2 named[909]: transfer of 'example.com/IN' from 192.168.64.10#53: connected using 192.168.64.10#53
Dec 10 10:41:44 ns2 named[909]: transfer of '0.0.10.in_addr.arpa/IN' from 192.168.64.10#53: connected using 192.168.64.10#53
Dec 10 10:41:44 ns2 named[909]: zone example.com/IN: transferred serial 2008111001
Dec 10 10:41:44 ns2 named[909]: transfer of 'example.com/IN' from 192.168.64.10#53: Transfer status: success
Dec 10 10:41:44 ns2 named[909]: transfer of 'example.com/IN' from 192.168.64.10#53: Transfer completed: 1 messages, 8 records, 215 byt
es, 0.003 secs (71666 bytes/sec) (serial 2008111001)
Dec 10 10:41:44 ns2 named[909]: zone 0.0.10.in_addr.arpa/IN: transferred serial 2008111001
Dec 10 10:41:44 ns2 named[909]: zone 0.0.10.in_addr.arpa/IN' from 192.168.64.10#53: Transfer status: success
Dec 10 10:41:44 ns2 named[909]: transfer of '0.0.10.in_addr.arpa/IN' from 192.168.64.10#53: Transfer status: success
Dec 10 10:41:44 ns2 named[909]: transfer of '0.0.10.in_addr.arpa/IN' from 192.168.64.10#53: Transfer status: success
Dec 10 10:41:44 ns2 named[909]: transfer of '0.0.10.in_addr.arpa/IN' from 192.168.64.10#53: Transfer status: success
Dec 10 10:41:44 ns2 named[909]: transfer of '0.0.10.in_addr.arpa/IN' from 192.168.64.10#53: Transfer completed: 1 messages, 6 records, 201 bytes, 0.003 secs (67000 bytes/sec) (serial 2008111001)



LICDAVIS

Step 3 Load balancing

- Tool: NGINX (an open-source software for load balancing)
- Load Balancing Algorithm: Least Connection Algorithm (LC)
- Parameters: proxy response limit=1; timeout duration=1s (If one server fails to respond within the specified time or query limit, the load balancer automatically redirects the request to the next available server)

	U II < Linux_Primary		••• U I <	Linux_Secondary2	0.01	● () < Linux_Seconda	ry4	₩ 5 Q @	
Activ	ities 🗈 Terminal	Dec 11 10:06	Activities 🗈 Te	minal Dec 11 10:06	Activi	ities 🗈 Terminal	Dec 11 10:06	í i	き む
6	P	root@ns0: /var/log	🝅 🖻	root@ns2: /var/log	6	n	root@ns4: /var/log		ø ×
	Dec 11 09:32:18 ns0 named[3] example.com; euery: www.exa Dec 11 09:32:18 ns0 named[3] example.com; euery: www.exa Dec 11 09:32:18 ns0 named[3] example.com; euery: www.exa Dec 11 09:32:18 ns0 named[3] Dec 11 09:32:18 ns0 named[3] Dec 11 09:32:18 ns0 named[3] example.com; euery: www.exa Dec 11 09:32:18 ns0 named[3] example.com; euery: www.exa Dec 11 09:32:18 ns0 named[3] example.com; euery: www.exa Dec 11 09:32:18 ns0 named[3] example.com; euery: www.exa	$\begin{array}{llllllllllllllllllllllllllllllllllll$	Dec 10 09: example.cc Dec 10 09: example.cc	22:25 nas2 named[907]: cltent @extfffec0. nb; quer; www.exanplc.com IN A +f(3)K (22:30 nas2 named[907]: cltent @extfff9002 nb; quer; www.exanplc.com IN A +f(3)K (23:305 nas2 named[907]: cltent @extff9002 nb; query; www.exanplc.com IN A +f(3)K (23:305 nas2 named[907]: cltent @extff9002 nb; query; www.exanplc.com IN A +f(3)K (23:305 nas2 named[907]: cltent @extff9002 20:57 nas2 named[907]: cltent @ext	8Cfi 192 4a6i 192 4a6i 192 4a6i 192 8Cfi 192 2aei 192 192 e38i 192 2aei 192 192	Dec 10 09:21:50 has hat example.com): guer; w Dec 10 09:21:50 has hat example.com): guer; w Dec 10 09:22:00 has hat example.com): guer; w Dec 10 09:22:05 has hat example.com): guer; w Dec 10 09:22:25 has hat example.com): guer; w	wed[906]: cluent gentrff727b06kg wecangle.com IN a \pm [60]K (192. wecangle.com IN a \pm [60]K (192.	192.168.64.2745786 198.64.14) 192.168.64.2740503 198.64.14) 192.168.64.2743662 198.64.14) 192.168.64.2743662 198.64.14) 192.168.64.2750127 168.64.2750127 198.64.42750127 198.7550127 198.7550127 198.7550127 198.7550127 198.7550127 198.7550127 198.7550127 198.7550127 198.755012	5 (WWW. 3 (WWW. 2 (WWW. 5 (WWW. 7 (WWW. 3 (WWW. 8 (WWW. 9 (WWW.
	● 🕛 🗏 🖾 Linux_Secondary1		••• U I 4	Linux_Secondary3		● 🕛 🗏 ⊲ Linux_Seconda	ry5	4 5 0 Ø	C" ©
Activ	ities 🗈 Terminal	Dec 11 10:06	Activities 🗈 Ter	minal Dec 11 10:06	Activi	ities 🗈 Terminal	Dec 11 10:06	,	き (1)
6	я	root@ns1: /var/log	📥 🗩	root@ns3: /var/log		Ē	root@ns5: /var/log		ø x
	Det 10 09:21:45 ns1 named[90 example.com): query: iwwe.exa bec: 10 09:21:45 ns1 named[90 example.com): query: iwwe.exa bec: 10 09:22:25 ns1 named[90 example.com): query: iwwe.exa bec: 10 09:22:35 ns1 named[90 example.com): query: iwwe.exa bec: 10 09:22:35 ns1 named[90 example.com): query: iwwe.exa bec: 10 09:22:35 ns1 named[90 example.com): query: iwwe.exa	1]: client g0xffff8c00f9bs mple.con IN A \pm [0]X (192.1) 1]: client g0xfff54032c28 mple.con IN A \pm [0]X (192.1) 1]: client g0xfff58063c38 mple.con IN A \pm [0]X (192.1) 1]: client g0xfff58063c38 mple.con IN A \pm [0]X (192.1) 1]: client g0xfff580675b8 mple.con IN A \pm [0]X (192.1) 1]: client g0xfff58063c38	Completes Dec 10 09: example.co Dec 10 09: example.co Dec 10 09: example.co Dec 10 09: example.co Dec 10 09: example.co Dec 10 09: example.co Dec 10 09:	22:40 nr3 named[917]: cltent g0xffff700 n); gury; www.example.con IN A + E(0)K (22:40 nr3 named[917]: cltent g0xffff000 n); gury; www.example.con IN A + E(0)K (22:45 nr3 named[917]: cltent g0xffff7000 n); gury; www.example.con IN A + E(0)K (22:45 nr3 named[917]: cltent g0xffff7000 n); gury; www.example.con IN A + E(0)K (22:45 nr3 named[917]: cltent g0xffff7000 n); gury; www.example.con IN A + E(0)K (22:45 nr3 named[917]: cltent g0xffff7000 n); gury; www.example.con IN A + E(0)K (32-1. 3eb8 192. 31b8 192. 3848. 192. 3848. 192. 31b8 192. 31b8 192. 31b8	Dec 10 09:22:55 ns5 nar example.com): query: w Dec 10 09:22:55 ns5 nar example.com): query: w Dec 10 09:23:00 ns5 nar example.com): query: w Dec 10 09:26:32 ns5 nar example.com): query: w Dec 10 09:26:42 ns5 nar example.com): query: w Dec 10 09:27:07 ns5 nar	<pre>med[902]: client @0xfff90270bf8 w.exaple.com IM A #{[0]K (152. ed[902]: client @0xfff900ib080 w.exaple.com IM A #{[0]K (152. ed[902]: client @0xfff900ib080</pre>	192.168.64.2#52412 108.64.15) 192.168.64.2#55360 168.64.15) 192.168.64.2#55619 168.64.15) 192.168.64.2#59788 168.64.5) 192.168.64.2#37061 168.64.15) 192.168.64.2#3326	2 (www. 3 (www. 9 (www. 8 (www. 1 (www. 5 (www.

UCDAVIS

Step 4 Evaluation

- Limitation: Due to our limited resources, genuine DoS attack is infeasible
- Alternative Solution: Send high volume of queries at one time for testing
- Parameters: Number of processes=5000; Number of total request=50,000
 - Scenario 1: Single server;
 - Scenario 2: Two servers with load balancing;
 - Scenarios 3: Six servers with load balancing.

Scenario	Number of Requests	Number of Answers	Respond Rate
1	50000	49551	99.1%
2	50000	49968	99.9%
3	50000	50000	100.0%

Table 2: Respond Rate Under Different Scenarios

Step 4 Evaluation

Open Y IFI	5000_50000_1.txt ~/Downloads	Save = - • ×	
1 Sun Dec 10 10:26:57 PM UTC 2023	Q 10.0.0.7	🖾 1 of 49551 🔿 🗸	
3; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> www.exa	ample.com		
5;; Got answer:			
6 ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53 7 :: flags: gr aa rd ra: OUERY: 1. ANSWER: 1. AUTHORITY:	0. ADDITIONAL: 1		
8			
10; EDNS: version: 0, flags:; udp: 1232			
11; COOKIE: 34f95a40d84d7fef0100000065763b32059ad13e4cf80 12:: OUESTION SECTION:	deee (good)		
13 ;www.example.com. IN A			
14 15;; ANSWER SECTION:			
16 www.example.com. 259200 IN A 10.0.0	. 7		
Open ~ A log	_5000_50000_2.txt ~/Downloads	Save = o x	
1 Sun Dec 10 10:31:58 PM UTC 2023	0 10007	图 1 of 49968 个 ~	
2 3; <<>> DiG 9.18.12-0ubuntu0.22.04.3-Ubuntu <<>> www.e>	kample.com		
4 ;; global options: +cmd			
6;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36	5773		
7 ;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 8	0, ADDITIONAL: 1		
9;; OPT PSEUDOSECTION:			
10; EDNS: Version: 0, Flags:; 00p: 1232 11; COOKIE: ef72612ecb1b76260100000065763c5e7e632f53fb56	020c9 (good)		
12;; QUESTION SECTION: 13:www.example.com. IN A			
14			
16 www.example.com. 259200 IN A 10.0.6	0.7		
Open ∽ ⊣	5000_50000_6.txt	Save = - o ×	
1 Sun Dec 10 10:53:54 PM UTC 2023			
2	2 10.0.0.7	(¥ 1 or 50000 ~ ~	
4 ;; global options: +cmd			
<pre>5 ;; Got answer: 6 :: ->>HEADER<<- opcode: OUERY. status: NOERROR. id: 31</pre>	686		
7;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY:	0, ADDITIONAL: 1		
9;; OPT PSEUDOSECTION:			
10; EDNS: version: 0, flags:; udp: 1232	f6hf (good)		
12 ;; QUESTION SECTION:			UCDAVIS
13 ;www.example.com. IN A	I		
15;; ANSWER SECTION:	7		22





Conclusion



Limitations and Future Work

Our work:

- We developed a DNS system on multiple virtual machines with fault tolerance;
- Our system includes features of automatic self-replication and an advanced mechanism of load balancing;
- In addition, we simulated an attack with a high volume of queries on our system, and the experiment result shows the robustness of our system in this scenario.

Future work:

- Optimize the load balancing algorithm and configuration to achieve better resilience;
- Test the results of the system more systematically given more resources;
- Consider other factors like data security.



Thank You

