PAPER Special Section on Information and Communication System Security

UDP Large-Payload Capability Detection for DNSSEC

Kenji RIKITAKE[†a)], **Koji NAKAO**[†,††], **Shinji SHIMOJO**[†††], ***and*** **Hiroki NOGAWA**[††††], ***Members***

SUMMARY Domain Name System (DNS) is a major target for the network security attacks due to the weak authentication. A security extension DNSSEC has been proposed to introduce the public-key authentication, but it is still on the deployment phase. DNSSEC assumes IP fragmentation allowance for exchange of its messages over UDP large payloads. IP fragments are often blocked on network packet filters for administrative reasons, and the blockage may prevent fast exchange of DNSSEC messages. In this paper, we propose a scheme to detect the UDP large-payload transfer capability between two DNSSEC hosts. The proposed detection scheme does not require new protocol elements of DNS and DNSSEC, so it is applicable by solely modifying the application software and configuration. The scheme allows faster capability detection to probe the end-to-end communication capability between two DNS hosts by transferring a large UDP DNS message. The DNS software can choose the maximum transmission unit (MTU) on the application level using the probed detection results. Implementation test results show that the proposed scheme shortens the detection and transition time on fragment-blocked transports.

key words: *DNS, DNSSEC, IP fragmentation, application MTU discovery*

1. Introduction

Domain Name System (DNS) [1]–[3] has been one of the required subsystems of the Internet. Maintaining the integrity and genuineness of each resource records (RRs) becomes more crucial for guaranteeing the Internet as a safe and secure place for human activities.

The traditional DNS has no mechanism to assure the authenticity of the RRs other than the weak authentication of verifying the delegation path of authoritative name servers. Attacks by forged DNS RRs become popular and cause social and financial damages.

Many DNS extension protocols have been proposed to amend the weakness of the traditional DNS. DNSSEC [4]–[6] is intended to provide cryptographic authentication of an resource record set (RRset), which is a set of RRs with the same label, class and type, but with different data [3]. DNSSEC ensures the authorization path of representing the hierarchy of the global DNS zone delegation.

Manuscript received July 30, 2007.

Manuscript revised December 12, 2007.

[†]The authors are with Network Security Incident Response Group, National Institute of Information and Communications Technology, Koganei-shi, 184–8795 Japan.

[††]The author is with KDDI Corporation, Tokyo, 102–8460 Japan.

[†††]The author is with Cybermedia Center, Osaka University, Ibaraki-shi, 567–0047 Japan.

[††††]The author is with Information Center for Medical Sciences, Tokyo Medical and Dental University, Tokyo, 113–8510 Japan.

a) E-mail: rikitake@nict.go.jp

DOI: 10.1093/ietisy/e91–d.5.1261

EDNS0 [7] is another extension of the UDP transport usage of DNS, which enables the exchange of larger payloads over UDP datagrams. The traditional DNS has the limit of 512 bytes for each UDP message (RFC1035 [2] Sect. 4.2.1). EDNS0 allows DNS software to specify the maximum size for each DNS message exceeding 512 bytes.

DNSSEC requires EDNS0 support, since DNSSEC needs to exchange large-size messages exceeding the maximum transmission unit (MTU) of lower-level protocols. A DNSSEC-aware name server should support a message size of 4000 octets [8].

MTU size values vary among the media and protocols. IP version 4 (IPv4) MTU over Ethernet is 1500 octets (RFC1122 [9] Sect. 2.3.3). IP version 6 (IPv6) [10] requires the link MTU of 1280 octets or greater. If the payload length of an IP packet exceeds those values, the datagram must be sent over multiple IP packets called IP fragments.

The implications of IP fragmentation are complex and cause various problems. The receiver of fragments must wait for the arrival of all fragments to reassemble them into the original datagram. Loss of a single fragment requires the upper-level protocol to retransmit the whole original datagram again [11]. Attacks with IP fragments to rewrite TCP headers are known as *Tiny Fragment Attack* and the protection against the attack has been proposed [12], [13].

While IP backbone networks do not block IP fragments, many end-user sites block the IP fragments as the default setting [14] for preventing unwanted security consequences. Some packet filters on firewall devices prevent traversal of large-packet and EDNS0-enabled DNS UDP datagrams [15]. This blockage degrades the performance of DNSSEC and other large-payload DNS message exchanges, which assumes the capability of UDP large-payload exchange.

When IP fragments are blocked between the DNSSEC-aware resolvers and servers, a typical workaround for the resolver is to wait for the UDP response at least for a few seconds and retry by sending a request in TCP. This fallback sequence will slow the response of an application and degrade the network usability. An active detection of UDP large-payload capability for the resolvers can reduce the fallback time and improve the usability.

In this paper, we propose a scheme to detect the UDP large-payload transfer capability between two DNSSEC hosts, over the DNS application protocol itself. This scheme does not require new protocol elements of DNS and DNSSEC, and applicable by solely modifying the applica-

tion software and configuration. The scheme allows faster capability detection to probe the end-to-end communication capability for each direction between a resolver and a server, and find out the appropriate UDP MTU values without any help of other MTU-discovery methods. In later sections, we first discuss the general transport requirement imposed by DNS extensions in Sect. 2, and analyze the interaction of IP fragmentation and DNS UDP transport in Sect. 3. In Sect. 4, we show the results of technical considerations and preliminary test results for the design of the application-level detection scheme for DNSSEC. In Sect. 5, we propose the scheme in details and show the evaluation test results. Our conclusion and further works are presented in Sect. 6.

2. DNS Extensions and the Transport Requirement

The length of messages exchanged between DNS resolvers and servers has a trend to increase, as more applications rely on DNS RRs and extensions. The emerging Internet applications require DNS to be sufficiently reliable for exchanging larger messages exceeding the traditional limit of 512 bytes, and even exceeding the limitation imposed by IPv4 and IPv6 MTUs.

Introduction of IPv6 and DNSSEC will require all end-hosts, intermediate routers, and packet filters to be able to exchange IP fragments generated by the upper-layer UDP. While DNS resolvers should support the TCP transport (RFC1123 [16] 6.1.3.2), still UDP is preferred because of the lower overhead, both in packet count and in server-side connection-state keeping. Fallback to TCP takes an overhead of timeout period by the resolver to detect the failure of UDP query, and it should be avoided as possible.

In this section, we describe the examples of the emerging applications and how they affect the requirements of DNS transport.

2.1 IPv6

IPv6 uses AAAA RR to refer an IPv6 address from a domain name, and `ip6.arpa` for the reverse lookups from an IPv6 address to a corresponding domain name [17].

When an A RR (RFC1035 Sect. 3.4.1) for an IPv4 address is replaced by an AAAA RR, the length increases by 12 bytes because of the address-length increase from 32 bits to 128 bits (4 to 16 bytes). During the IPv4-to-IPv6 transition, adding an AAAA RR to an existing A RR for a host costs at least 28 more bytes, assuming the domain-name compression enabled (RFC1035 Sect. 4.1.4).

In our previous works [18]–[20], they showed a simulation result based on a real-world traffic data that the percentage of DNS answers exceeding the 512-byte UDP payload size limit of the traditional DNS, including the additional records, increased from 0.04% to 1∼3% during and after the migration from IPv4 to IPv6. As more and more hosts are likely to use IPv6, the DNS answers will have a trend of continuously increasing length.

2.2 DNSSEC

DNSSEC uses its own RRs [5]. DNSSEC-aware servers authenticate each RRset by adding an RRSIG RR as the signature resource record. Each RRSIG RR includes a 128-byte signature of RSA/SHA1 [21]. DNSKEY RRs of 514-byte Key-Signing Key (KSK) and 130-byte ZSK (Zone-Signing Key) are used for providing the RSA/SHA1 public keys for the signatures.

Two new types of RRs are also added; Next Secure (NSEC) to ensure the authenticated denial of existence of a non-existent query, and Delegation Signer (DS) to show the DNSKEY RR used in the authentication process traversing the zones.

The answering action for a denial-of-existence with a Name Error (NXdomain) response (RFC1035 Sect. 4.1.1) is also changed by the introduction of DNSSEC. The traditional non-DNSSEC response only requires sending an answer without additional RRs with RCODE=3 response. The NXdomain response for DNSSEC requires sending the Start-of-Authority (SOA) RR of the zone, two NSEC RRs with their RRSIGs, and the DNSKEY RRs with the RRSIGs [22].

New RRs introduced by DNSSEC largely contribute to increase the message size of DNS. In one of our previous works [23], we conclude that 30% of the payloads will not fit in an IPv6 packet of 1280-byte MTU without fragmentation, by a payload-length estimation with a real-world traffic data. Ager et. al [22] also report that the size of 72∼77% of the DNSSEC answer payloads including Name Error (NXDomain) responses will exceed the IPv4 fragmentation limit.

2.3 Other Extensions and Applications

DomainKeys Identified Mail (DKIM) [24] uses DNS Text (TXT) RR, to verify the signature attached for each signed messages. Sender Policy Framework (SPF) [25] uses an SPF RR and TXT RR, to declare authorized hosts as a part of sender identities for SMTP [26] mail transmission.

SSHFP RR [27] is proposed to store Secure Shell (SSH) [28] fingerprints. IPSECKEY RR [29] is proposed to store the public keys for IPsec [30]. CERT RR [31] is proposed to store X.509 certificates and certificate revocation lists.

SRV RR [32] is proposed to provide information of multiple hosts designated for a specific network service. NAPTR RR is a part of Dynamic Delegation Discovery System [33]–[36], for identifying available services of an E.164 number [37], such as a telephone number.

DHCID RR [38] is a part of the resolution scheme of domain name conflicts between the Dynamic Host Configuration Protocol (DHCP) [39], [40] clients [41]–[43], so that the mapping between dynamically-assigned IP addresses and domain names are promptly updated through the authoritative DNS servers through the DNS UPDATE [44] protocol.

3. DNS UDP Transport and IP Fragmentation

3.1 IP Fragmentation and Large-Payload UDP Usage

Delivery of large UDP datagrams exceeding the MTU assumes the end-node fragmentation of a UDP datagram into multiple IP packets. Many wide-area UDP application try to avoid IP fragmentation by restricting the UDP payload size. For example, SIP [45], a signaling protocol for Internet telephony, mandates a larger request to be handled over TCP. Real-time Transport Protocol (RTP) [46] specifies that the control protocol packets should be segmented into multiple shorter ones so that they will not exceed the MTU of the path. Comparing to those newer protocols, DNS is an old application which solely depends on the IP fragmentation for the large-payload UDP delivery.

IP fragmentation may degrade the reliability of UDP datagram delivery. Theoretically, a datagram split into fragments is more prone to errors. When the failure rate of a fragment delivery is p, the failure rate of the overall datagram of n fragments is $1 - (1 - p)^n$, which is $\simeq np$ when p is very small. We conclude, however, the UDP datagram delivery rate will not be largely changed by the size increase and the fragmentation, at least from an end-user point of view. Our measurement results on an end-to-end UDP delivery over small operational networks [47] shows that no correlation between the payload length and the loss rate of the transferred UDP datagrams, when the bandwidth congestion does not occur on each side of the transfer. The results suggests that IP fragmentation itself will not largely affect the overall reliability of large-payload DNS message delivery.

3.2 End-Node IP Fragments and Packet Filters

Protecting Internet nodes from unwanted intrusion and exploit by filtering out suspicious packets has been a common practice. For many end users, filtering out IP fragments is a *default* practice for implementing packet filters, since most of the services over Internet still work well without IP fragments, if TCP, non-fragment UDP, and ICMP packets get through the packet filters between two end nodes.

Most of the popular Internet services, such as Web and e-mail, are based on TCP. We observe all the IP packets from a FreeBSD [48] server contained the TCP payload are with the *don't fragment* (DF) bit set. The DF-set packet usage of TCP is an expected behavior of hosts enabled the Path MTU discovery [49], [50]. TCP is a sliding-window protocol and the maximum segment size can be set below the IP MTU to avoid fragmentation. Two major UDP services, traditional DNS and Network Time Protocol (NTP) [51], do not require IP fragments, since the UDP payload size is smaller than minimum MTU of 576 bytes (RFC1122 Sect. 3.3.2). Most of the ICMP control packets do not require IP fragmentation so long as the payload size is smaller than the link MTU.

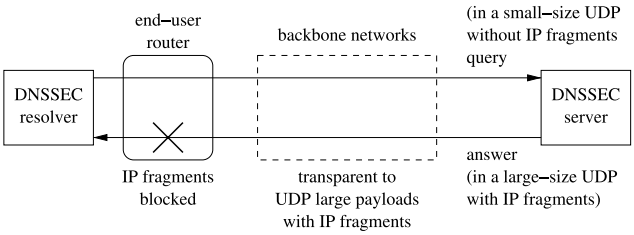

Fig. 1 DNS answers blocked by the end-user router.

Table 1 Common examples of UDP payload size values for EDNS0.

Size (bytes)	Description of the size value
512	Traditional DNS limit, also chosen on EDNS0 of BIND 9.4.2 as a value for preventing firewall blockage
1232	Maximum size for IPv6 (default MTU: 1280 bytes)
1280	Suggested in EDNS0 [7], also used in Windows Server 2003
1472	Maximum size for IPv4 (default MTU: 1500 bytes)
2048	Default size for BIND `dig`, `host` and `nslookup` commands, before the version 9.4.0b3
4096	Default size for BIND 9.4.2 libdns resolver library

DNSSEC, however, will *not* work if IP fragments are blocked by the packet filters in between the two end nodes. We experience the adverse effect of filtering against DNSSEC [14], which blocks them from reaching a DNSSEC server by UDP, while testing the reachability of a campus network to the DNSSEC-enabled registry service of NIC-SE [52], the registrar for the `.se` country top level domain. We have later learned that backbone network providers do not block the fragments at all, and that the packet filter of the router which connects the end node used for the test actually prohibits the IP fragments to get through. This situation (Fig. 1) can also be caused by a firewall device incompatible with EDNS0 or UDP large payloads [15].

We have also surveyed common examples of UDP payload size values of EDNS0 in Table 1 (Rikitake [20], Sect. 3.5.2). The values suggest that the range of 512∼4096 bytes should be able to be chosen for the EDNS0 UDP payload size for DNSSEC.

3.3 Summary of This Section

In this section, we have shown the importance to assess the feasibility of DNSSEC is to ensure the bidirectional end-to-end capability of large-size DNS UDP payloads exceeding the traditional 512-byte limit, for the following reasons:

- IP fragments do not necessarily get through the packet filters on all operational networks, especially on those of the end-user networks;
- Large-payload datagram transmission is required for full transmission of DNSKEY RRs and the related signature RRs for the public-key delivery from authoritative servers; and
- UDP transport is preferred for DNS communication to

reduce the load of DNS servers and caches of memorizing TCP connection states and to avoid exchanging larger number of packets.

4. Considerations of UDP Large-Payload Detection Scheme for DNSSEC and the Preliminary Test Results

In this section, we present the details of preliminary test results for our proposed scheme. The scheme is based on a scheme exchanging a large-payload UDP datagram between DNS resolvers and servers for the first time of communication to determine the capability. We need to determine whether a given size of UDP payload succeeds or fails to pass for each of the two directions (Fig. 2) between the resolver and the server.

We conduct some preliminary tests to find out the possible methods for sending and receiving a large-payload UDP datagram, in a pre-defined size, exceeding 512 bytes, and the path MTU between the resolvers and servers. We have tested two ideas of large-payload UDP delivery for DNS, for the server-to-resolver and resolver-to-server directions. We conclude that the resolver-to-server direction should be chosen in our proposed scheme, to determine whether an end-to-end large-payload UDP delivery for a given size is possible.

We first describe what kind of DNS specification documents should be considered in Sect. 4.1. We then describe the preliminary test overview of the server-to-resolver test in Sect. 4.2, and the resolver-to-server test in Sect. 4.3. We then compare the two methods regarding the operational network consideration in Sect. 4.4, and the EDNS0 compliance of the queries in Sect. 4.5. We describe the details of the preliminary test in Sect. 4.6, and the source code analysis of the related part of DNS programs in Sect. 4.7.

4.1 DNS Application Protocol Specification

We define *the DNS application protocol* in later sections as described in the protocol documents of RFC1034 [1], RFC1035 [2], and EDNS0 (RFC2671 [7]). The related sections of the RFC documents are:

- The form of DNS RRs (RFC1035 Sect. 3)
- The form of DNS message format (RFC1035 Sect. 4)
- EDNS0 specification (RFC2671), which allows *none* or *one* OPT RR in the each DNS message (query/answer)

DNS query is defined to have only question sections, no answer/authority/additional section, and an optional OPT RR. In major implementations such as BIND [53], NSD [54] and djbdns [55], only one question is allowed for each query message.

EDNS0 allows the sender to add either zero or one OPT RR at the additional data section. The OPT RR has the fixed part (Table 2) and an optional variable part (Table 3). The fixed part contains the sender's UDP payload size, which shows the maximum acceptable size of UDP datagram.

The variable part of OPT RR allows multiple elements of the options. The option code must be registered to Internet Assigned Numbers Authority (IANA) for an official use [56]. An example of the usage of the variable part is to send and receive DNS Name Server Identifier (NSID) Option [57].

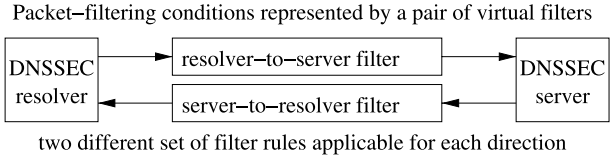

Fig. 2 Packet-filtering conditions have to be determined for each direction between the resolver and server.

Table 2 The overall data structure of OPT RR.

Fixed part (length: 11 bytes)		
Field name	Length (bytes)	Description
Name	1	empty (Root Domain label)
Type	2	value: decimal 41
Class	2	sender's UDP payload size
TTL	4	extended RCODE and flags
RDLEN	2	total length of the variable part
Variable part (optional)		
RDATA	(described in Table 3)	

Table 3 The variable part data structure of OPT RR.

Variable part elements (zero or more elements allowed in the variable part)		
OPTION-CODE	2	Option code for each element
OPTION-LENGTH	2	Option length of the option-data field for this element
OPTION-DATA	(*)	Variable-length data (*): length in OPTION-LENGTH (the meaning of data is separately defined for each OPTION-CODE)

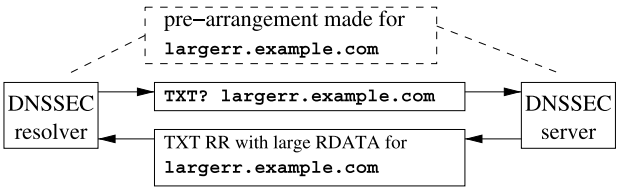

Fig. 3 Testing server-to-resolver direction path by requesting a pre-arranged large RR.

4.2 Requesting a Large RR

Sending a request for a large RR exceeding the path MTU size is a practical method to find out the server-to-resolver direction path MTU (Fig. 3), since a server cannot split an RR to send, so the contents will not be truncated. A resolver sends a request with a pre-defined name for the large RR, and the server returns it as the response.

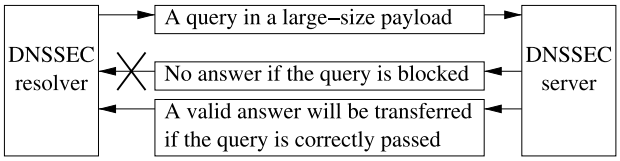

Fig. 4 Testing resolver-to-server direction path by sending a large query.

A TXT RR is a good candidate for the testing, since the zone entry for the RDATA field of the RR can be created with concatenating *character-strings*, a set of up-to-255-byte-each text string with the 1-byte length octet attached (RFC1035 Sects. 3.3 and 3.3.14). For example, the zone compiler utility of NSD [54] version 3.0.7, can create a large-size TXT RR record concatenating 64 label strings, which results in up to 16384 bytes in the RDATA field, well above the size we needed.

We have a successful result on requesting and obtaining a TXT RR of 2550 bytes [47] between a resolver and an authoritative server of BIND [53], over the networks of operational wide-area Internet.

4.3 Sending a Large Query

Sending a large UDP datagram exceeding the path MTU as a query to a DNS server is another practical method to find out the capability for the server to receive a large UDP query (Fig. 4), since the UDP message will not be processed by the server if the reassembly of related IP fragments fail.

We test this idea of making a 2048-byte payload query by including an arbitrary set of bytes in a message after an EDNS0-compliant query for the NS RRs of the Root Zone [47]. This query works fine without problem. We have also tested sending an incomplete IP datagram with *more fragments* (MF) bit set, and confirms that an incomplete is not accepted. We conclude that DNS servers in general are highly likely to safely ignore the byte strings after a legitimately-formed query.

We should note that in this experimentation [47], the arbitrary set of bytes *does not* conform to the EDNS0 variable part format, which is later found a source of problem, during the source code analysis as shown in Sect. 4.7. We have revised this idea by sending the string which has one variable option element of EDNS0, as later described in Sect. 4.5, to conform to the EDNS0 specification.

4.4 Considerations on Operational Networks

Various kinds of DNS software are running on the operational networks on Internet. The method in Sect. 4.2 requires to make a pre-arrangement for known domain name between the resolvers and servers. This means the resolvers to use the method have to know the domain name to let the server send TXT RRs before sending the request.

The method in Sect. 4.2 has another downside of possible exploitation for a denial-of-service (DoS) attack for a malicious third party, just by requesting the large RR. This

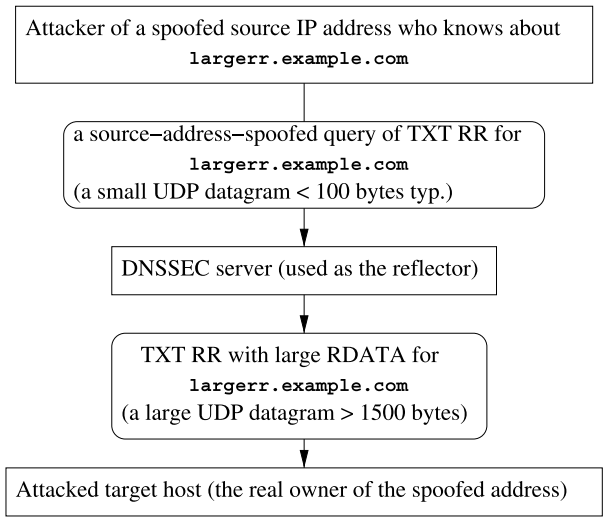

Fig. 5 DoS attack scenario by exploiting a large RR.

leads into a attacking scenario (Fig. 5) similar as the DNS Amplification Attack [58]. In this scenario, an attacker uses a small query packet to generate a large attack packet as the answer, of the large TXT RR of `largerr.example.com`. The attacker can redirect the TXT RR to an arbitrary target, by sending a query with the spoofed IP source address of the target.

We conclude that the method will not be widely deployable on the operational networks, since we consider that the attacker is highly likely to find the TXT RRs to exploit for this attack scenario, for the following reasons:

- the domain name for the TXT RRs cannot be kept secret to an attacker, without using a pre-exchanged secret piece of information between the resolver and server;
- an attacker and a legitimate resolver uses the same method to obtain the TXT RRs from the server, and there is no way to find out the legitimacy of query requests; and
- using a pre-exchanged secret piece of information between large numbers of hosts is expensive and difficult to deploy.

We also find a uni-directional test of the resolver-to-server direction is sufficient to find out whether the resolver-server DNS transactions can continue on the end-to-end fragmented UDP datagrams.

As shown in Fig. 3, the server must correctly receive the query message, to send the answer back to the resolver. If the resolver sends a large UDP query message with multiple IP fragments, the server *must* receive *all* the IP fragments of the UDP datagram. The same principle can be also applied to the server-to-resolver UDP datagram delivery.

In short, the bidirectional IP fragment delivery must be guaranteed to complete the query-and-answer transaction, if the resolver sends a large UDP query with multiple IP fragments, and if the server sends a large UDP answer with

Table 4 Expected status of answer messages when UDP query with multiple IP fragments are sent from the resolver to the server.

Status of IP fragmentation		Expected status of answer messages in multiple transactions
Sender-to-receiver-	Receiver-to-sender-	
Passed	Passed	The transactions continue without timeouts
Passed	Blocked	The transactions continue without timeouts so long as the answer messages do not fragment (i.e., the IP packets are smaller than MTU); the timeout occurs when the answer messages fragments, so the resolver can fallback to avoid IP fragmentation
Blocked	Passed	The transaction will not happen since the queries are blocked; this can be detected by the timeout
Blocked	Blocked	The transaction will not happen since the queries are blocked; this can be detected by the timeout

multiple IP fragments. This case can be practically detected on the resolver by using the timeout timer to wait for the response from the server.

Theoretically, there may be a case that the resolver-to-server IP fragment delivery is allowed, but the server-to-resolver one is *not* allowed. Sending a large UDP query with multiple IP fragments will not be able to specifically detect this case, since the resolver will receive the answer if the answer does not fragment into multiple IP packets. We do not have to *specifically* detect this case, however; so long as the resolver-server transactions continue, the resolver does not have to intervene. If the timeout happens, the resolver can change the operation mode to avoid the IP fragmentation by reducing the EDNS0 UDP size in the query packets under the MTU (Table 4).

Moreover, in the operational networks, DNSSEC server host is highly unlikely to prohibit IP fragments to send and receive regarding the operational requirements of DNSSEC, since the server host must be able to exchange UDP-based DNSKEY RRs with other authoritative DNSSEC servers over UDP for the minimal communication overhead. Under such a condition, we consider performing a uni-directional large-payload test of the resolver-to-server direction is sufficient to determine the capability between a resolver and a server.

Regarding the considerations described in this section, we conclude to focus on the method in Sect. 4.3 in our scheme. The method also has the following advantages over the one in Sect. 4.2:

- the method does not require a pre-arrangement between the resolvers and servers;
- the attacking scenario shown in Fig. 5 is not effective since the answer size can be set much smaller than the query size; and
- the resolver can measure the large-payload capability in details by controlling the size of the query.

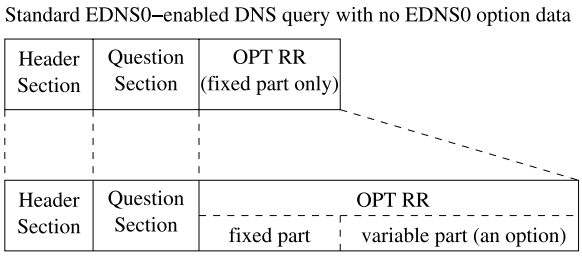

Fig. 6 DNS message format of EDNS0-compliant large queries.

4.5 Sending EDNS0-Compliant Large Queries

In the method in Sect. 4.3, the DNS message used does not conform to the EDNS0 specification. No detailed malfunction test is performed either. Forming a large-size UDP DNS query should be done to avoid causing malfunction of the actual programs and to comply with the related standards.

Figure 6 shows the DNS query message format in our scheme. We use the variable part of OPT RR to add a variable length of byte string to send a given size of UDP payload, to make the message EDNS0-compliant. Messages formed in this way retain the functionalities of normal queries and answers, as well as the UDP payload size and resolver DNSSEC-support [59] notification function of the fixed part of OPT RRs. The total payload size is the sum of the header, the question section, and the OPT RR in the additional section.

4.6 Preliminary Evaluation Test and the Summary

We perform an implementation test between a query generator and the authoritative server and cache programs, to find out actual difference of response to the proposed scheme as described in Sects. 4.3 and 4.5. We use the following configuration of the subsystems:

- Net::DNS [60] resolver package for Perl [61] programming language for the faster prototyping;
- Two i386-architecture computers running FreeBSD 6.2-RELEASE operating system for the resolver and server, connected to a 100BASE-TX 100 Mbps Ethernet using IPv4;
- IP Filter [62] 4.1.13, a packet filter tool, bundled to the FreeBSD 6.2-RELEASE; and
- BIND 9.4.2, NSD 3.0.7 and djbdns [55] 1.05 for the evaluation target programs.

We use the same resolver software for evaluating the target authoritative server and cache programs. We use the option code of 65535 (hex 0xffff) for the OPT RRs sent to each server program, which is not an assigned value for other protocols [†].

[†] An IANA document [56] defines the option codes of $0 \sim 3$ have already been assigned.

Table 5 Preliminary test result summary for the target DNS programs.

Program	(*)	Test results and remarks
named	yes	returns correct answers for $n \leq 4096$
nsd	yes	FORMERR and the incomplete OPT RR without RDLEN and RDATA for $n \leq 4096$ (a patched version [63] returns correct answers)
tinydns	no	returns correct answers when $n \leq 512$, no answer when $n \geq 513$
dnscache	no	returns correct answers when $n \leq 1023$, no answer when $n \geq 1024$

(*): DNSSEC/EDNS0 support [yes/no]
n: UDP payload length [bytes]
Tested values of *n*: 512, 513, 1023, 1024, 1473, 2048, 4096
named is of BIND 9.4.2 / *nsd* is of NSD 3.0.7
tinydns and *dnscache* are of djbdns-1.05

We use the IP Filter software to block and allow the IP fragments between the resolver and server programs.

Table 5 shows the test results. The payload values are chosen to detect the difference by the IP fragmentation on the Ethernet link, and to show the behavior change for known threshold values for each tested program. The results suggest the following common behaviors:

- IP fragmentation do not change the behavior of each software package, so long as the UDP query is completely received.
- DNSSEC-supported programs *do* respond for all the tested payload length values.
- The djbdns programs, which do not support DNSSEC, do not respond for the payload length values which cause IP fragmentation.

4.7 Source Code Analysis

We have analyzed the source code of each tested programs and have found out the related C-language functions for the incoming OPT RRs. The following analyzed results are consistent with the behavior of the tested programs as shown in Table 5:

- BIND 9.4.2 *named* parses the OPT RR in the function `client_request()` of `bin/named/client.c`. The function checks if the EDNS version is 0. Another function called `fromwire_opt()` of `lib/dns/rdata/generic/opt_41.c`, which performs the conversion of the DNS message of OPT RR to the internal data structure, also performs the data boundary check of each element of the variable part of OPT RR. The software, however, does not care about the contents of each OPTION-DATA, or about the OPTION-CODE.
- NSD 3.0.7 *nsd* parses the OPT RR in the function `edns_parse_record()` of `edns.c`. This function returns with an error code to generate DNS error with FORMERR when the OPT RR has an option data, which is a bug preventing execution of the parse code for NSID. We have provided a fix [63] so that the OPT RR option data contents are properly parsed [†].
- djbdns-1.05 has two programs, *tinydns* as an authoritative server and *dnscache* as a cache. Neither program parses the incoming OPT RR and simply ignores it when received.
  dnscache has the UDP receiving buffer limit of 1024 bytes, hard-coded in function `u_new()` of `dnscache.c`. If the received UDP payload exceeds 1024 bytes the datagram is discarded.
  tinydns has the UDP receiving buffer limit of 513 bytes, hard-coded as a static array in `server.c`. If the received UDP payload exceeds 512 bytes the datagram is discarded.

5. An Implementation of UDP Large-Payload Detection Scheme for DNSSEC and the Test Results

5.1 DNS Application MTU Discovery and the Proposed Scheme

Application MTU discovery is effective in a case when ICMP for the IP-level path MTU discovery does not properly work, which is common due to a defensive packet filter setting. Packetization Layer Path MTU Discovery [64] is a generic method for TCP without the help of ICMP messages.

DNS has no mechanism to actively ensure the communication capability of UDP large payloads between the resolvers and servers. EDNS0 only declares the size of the payload which can accept by each host, so that the other party of communication can change the maximum size of the sending payload. Each DNS application software is responsible to ensure the maximum size of a payload acceptable on a resolver-server communication path.

DNS software handles the UDP large payload issues mostly by manual and static configurations. For example, BIND version 9.4.0 and later versions allow making manual configuration change to enable and disable DNSSEC validation function while a server is running, as well as limiting maximum EDNS0 payload size independently for sending and receiving on a server. The dependency on static and manual configuration, however, does not cover the capability difference between individual hosts, and may lead to performance degradation caused by a configuration error. A probing mechanism should be introduced to detect the capability of sending and receiving UDP large datagrams in two or more IP fragments, to manage the per-host individual capability difference and to automatically find out the optimal configuration.

For the actual implementation, the following points should be considered:

- Retrying the large-sized queries should be avoided since they consume more network bandwidth, especially on the server side where those queries are aggregated.

[†]This patch is originally for NSD 3.0.5, but the same patch is also applicable to NSD 3.0.7 as well.

- Well-known servers such as the Root Servers and the authoritative servers for the top-level domains are highly likely to accept EDNS0 and IP fragments, so probing to those servers should be excluded.
- Sending large-sized queries may be only periodically needed by cacheing the capability status for each resolver-server path. The cached capability information should be refreshed if it contradicts with the actual behavior of UDP queries and answers.

We should also note that a known vulnerability exists for BIND 8.3.x through 8.3.3, which allows remote attackers to cause a DoS (termination due to assertion failure) via a request for a subdomain that does not exist, *with an OPT resource record with a large UDP payload size* [65]. BIND 8.3.x is no longer a supported version, however, and the fixed versions have already been freely available, so we consider this vulnerability will not largely affect the feasibility of our proposal.

5.2 An Implementation Example on BIND 9.4.2 Resolver Code

An example of DNS Path MTU discovery with EDNS0 information has been implemented in BIND 9.4.2 resolver code in function `resquery_send()` of `lib/dns/resolver.c`. The resolver first tries the EDNS0 query with the UDP size of 4096 bytes as the default setting, retries a pre-defined timeout (3 seconds from the first query) to fall back to a 512-byte EDNS0 query (to relax the traditional restriction of 512-byte MTU for DNS), and to abandon the EDNS0 at all for the lookup if another timeout (6 seconds from the first query). This graceful fallback approach depends on relatively long timeouts and adds overall latency on a lookup. This approach does not consider the value of IP MTU size, so the EDNS0 UDP size are restricted to 512 bytes even if the IP MTU in-between the server and resolver may allow a larger message size value.

Figure 7 is an example of implementation with the proposed scheme, as a modified BIND 9.4.2 resolver code. This example changes the first query for a server by adding a one-element variable part of OPT RR so that the query will always cause IP fragmentation. If this large-UDP query fails, the modified code will try by falling back to an EDNS0 query with the UDP size of 1400 bytes, which will not cause IP fragmentation on IPv4 over Ethernet and PPPoE [66] links.

The test status of the large-UDP query and 1400-byte EDNS0 UDP size query are remembered for each destination host IP address, as the original BIND 9.4.2 code does for the EDNS0 denial status. Keeping the test status for each destination host solves the issues cited in proposed in Sect. 5.1.

The following modifications, other than the implementation of pseudo-code in Fig. 7, have been done on the BIND 9.4.2 source code:

- changing the value of `SCRATCHPAD_SIZE` in `lib/dns/message.c` from 512 to 4096, to accommodate a large-size UDP message; and
- changing the size of the `data` member of structure `query` in `lib/dns/resolver.c`, an `unsigned char` type array, from 512 to 4096 bytes to accommodate a large-size UDP query.

```
// The EDNS0 udpsize values are set for
// IPv4 Ethernet MTU (1500 bytes) and
// a typical PPPoE MTU (1454 bytes)
//
// *** our proposed scheme implementation begins here
//
if (a query with 1504-byte variable part of OPT RR
        is answered before the timeout (0.5 second)) {
        // IP fragments get through the path
        Continue later queries with IP-fragment-allowed EDNS0;
        Remember the fragment status for another query;
        /* exiting this if statement */
} else if (a query with 1400-byte EDNS0 UDP size
        is answered before the timeout (1 second)) {
        // 1400-byte UDP size EDNS0 works on the path, so use it
        Continue later queries with 1400-byte UDP size EDNS0;
        Remember the EDNS0 UDP size status for another query;
        /* exiting this if statement */
} else
//
// *** our proposed scheme implementation ends here
//
// *** the BIND 9.4.2 EDNS0 original fallback sequence begins here
// (Note that BIND 9.4.2 original code tries the query first
// with the default UDP size value (4096) value,
// which is not the case in our modified code)
//
if (a query with 512-byte EDNS0 UDP size)
        is answered before the timeout (3 seconds)) {
        // at least 512-byte EDNS0 query passed unfiltered
        Continue later queries with 512-byte EDNS0;
        /* exiting this if statement */
} else {
if (a query with non-EDNS0 UDP size
        is answered before the timeout (6 seconds)) {
        // at least non-EDNS0 query passed unfiltered
        Continue later queries without EDNS0 (traditional DNS);
        Remember the non-EDNS0 status for another query;
        /* exiting this if statement */
}
```

Fig. 7 A pseudo-code for detecting the fragment-blocked DNS path based on our implementation of the BIND 9.4.2 resolver code.

5.3 Evaluation Test Settings

We conduct the performance test using the modified code based on our scheme, as the cache server. We use the following configuration of the subsystems (Fig. 8):

- Two i386-architecture computers running FreeBSD 6.2-RELEASE operating system for the resolver and server, connected to a 100BASE-TX 100 Mbps Ethernet using IPv4;
- IP Filter [62] 4.1.13, a packet filter tool, bundled to the FreeBSD 6.2-RELEASE, for the IP fragmentation blocking and allowance, on the cache side of the hosts;
- *dummynet* [67], a traffic shaping and delay emulation

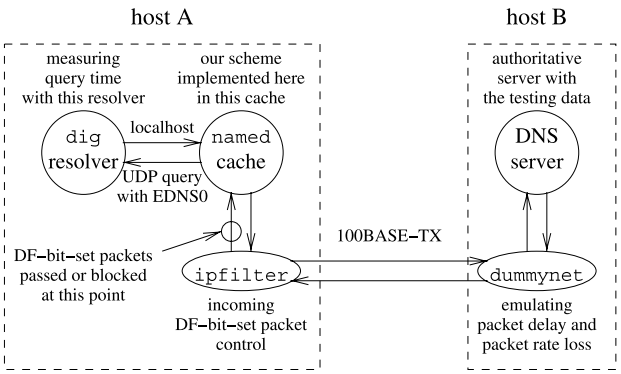

Fig. 8 Testing system configuration of our scheme.

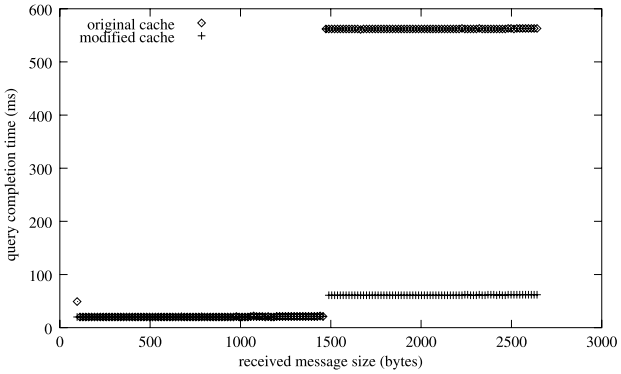

Fig. 9 Completion time of DNS queries on 10 ms-delay link with no additional packet error while incoming DF-bit-set packets are blocked on DNS cache.

tool, bundled to the FreeBSD 6.2-RELEASE, for the simulated packet loss and delay, on the authoritative server side of the hosts;
- *tcpdump* [68], a packet logging and analysis tool, bundled to the FreeBSD 6.2-RELEASE, to analyze the packet flow between the programs;
- original (unmodified) BIND 9.4.2 *named* as the authoritative server, which has the zone data for the testing traffic;
- original and modified BIND 9.4.2 *named* for the cache, which includes the resolver code being tested in the modified version; and
- the *dig* utility of BIND 9.3.3, bundled to the FreeBSD 6.2-RELEASE, for invoking the queries to the cache, and measure the query completion time [†].

We measure the query completion time for the 160 message size values by sequentially querying A RRs for domain names under `example.net` [††], which has 1 to 160 A RRs for each domain name defined [†††]. The actual message size of the answers are 96 ∼ 2642 bytes. The on-memory database of received RRs in the tested cache is emptied every time before the test begins, so that the results will not be affected.

The tests are conducted for the following simulated packet loss rates and delay times between the two hosts:

delay times: 10 ms and 100 ms
packet loss rates: zero (none added by *dummynet*), 1%, and 10%

We conduct the tests by comparing the following condition elements:

DNS cache: original (unmodified) and modified versions
IP fragment conditions: blocking and allowing incoming (server-to-cache) DF-bit-set IP fragments, by configuring the IP Filter of the host running the cache program

5.4 Evaluation Test Results

5.4.1 Cases of 10 ms Delay with No Added Packet Loss

We first compare the results of the cases of 10 ms delay and no added packet loss. When IP fragments are allowed, the query completion time values for all 160 queries were in the range of 20 ∼ 22 ms, in either case of original or modified cache. This shows that the modification will not impose a significant overhead when the link allows IP fragmentation.

Figure 9 shows the results of the cases of 10 ms delay and no added packet loss, but the IP fragments are blocked. When the answer packets do not fragment. the results are in the same range (≃ 20 ms) as in the cases when IP fragments are allowed. When the answer packets begins to fragment due to the large size (> 1472 bytes of UDP message), the query completion time values are increased to 562 ∼ 563 ms for the original cache. For the modified cache, the increased values are 61 ∼ 62 ms, except for the once when the answer size is 1473 bytes the completion time is 562 ms.

The difference of behavior between the original and modified caches shows that the original cache has to timeout once (for 500 ms) in addition to detect that the large-size UDP answers are blocked, while the modified cache has to timeout only once to detect the link status and use the information for later queries to prevent timeouts.

By analyzing the detailed *tcpdump* packet exchange logs between the cache and the authoritative server, we discover that the ≃ 60 ms query time for the modified cache is caused by two successive DNS exchanges of a UDP query with truncated answer and a TCP query, between the cache and the authoritative server [††††].

[†]The dig utility of BIND 9.3.3 was configured to allow up to 4096 bytes of UDP answer size, and wait up to 60 seconds for giving up the queries.

[††]The names used are `test0.example.net`, `test1.example.net`, ... `test159.example.net`, which the number after `test` are 0 ∼ 159.

[†††]The numbers of A RRs are 1 for `test0.example.net`, 80 for `test79.example.net` and 160 for `test159.example.net`; the number of A RRs are $(n + 1)$ when the number after `test` is n.

[††††]A TCP query takes 2 packet round trips in minimum (when no error occurs) to complete. A UDP query only needs one round trip.

Table 6 Statistics for the query completion times for different packet loss rates on 10 ms-delay links while incoming DF-bit-set packets are blocked at the cache.

Packet loss rate (%)	original cache		modified cache	
	μ	σ	μ	σ
(*) 0	271.09	270.85	42.33	46.12
1	313.90	342.30	44.00	51.19
10	826.41	1240.71	489.76	1124.64

μ: mean value (ms)
σ: unbiased standard deviation (ms)
(*): no error added by *dummynet*

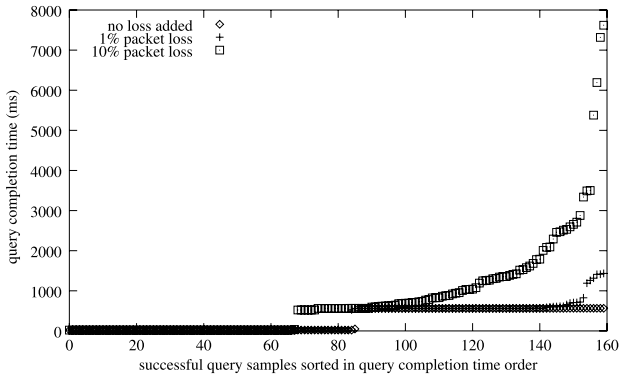

Fig. 10 Query completion time of DNS queries on 10 ms-delay link for different packet loss rates while incoming DF-bit-set packets are blocked on the original DNS cache.

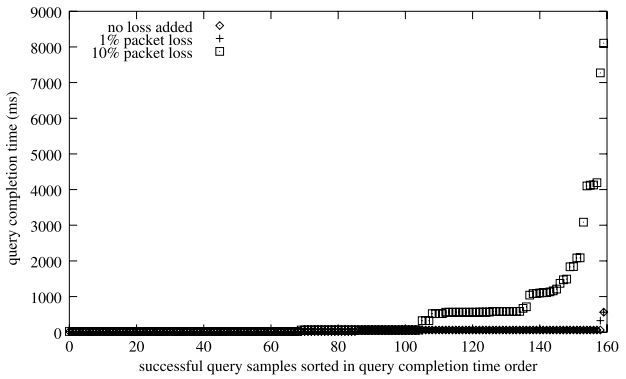

Fig. 11 Query completion time of DNS queries on 10 ms-delay link for different packet loss rates while incoming DF-bit-set packets are blocked at the modified DNS cache.

5.4.2 Cases for Different Packet Loss Rates

We compare the test results of the cases of different packet loss rates, for the 10 ms delay time, when IP fragments are blocked. Table 6 shows the statistics of the queries. Figure 10 shows the results of the cases on the original cache, and the Fig. 11 shows those on the modified cache.

In all packet loss rate cases, the modified cache gives shorter mean values of query completion time, especially when the loss rates are small. The overhead of the original cache, which requires one timeout for each query between the resolver and the cache, contributes the difference. The results show that the modified cache is more robust than the original cache against the packet loss between the cache and the authoritative server.

Table 7 Statistics for the query completion times for different delay times on 1% packet loss rate while incoming DF-bit-set packets are blocked at the cache.

Delay time (ms)	original cache		modified cache	
	μ	σ	μ	σ
10	313.90	342.30	44.00	51.19
100	913.33	791.12	415.15	240.98

μ: mean value (ms)
σ: unbiased standard deviation (ms)

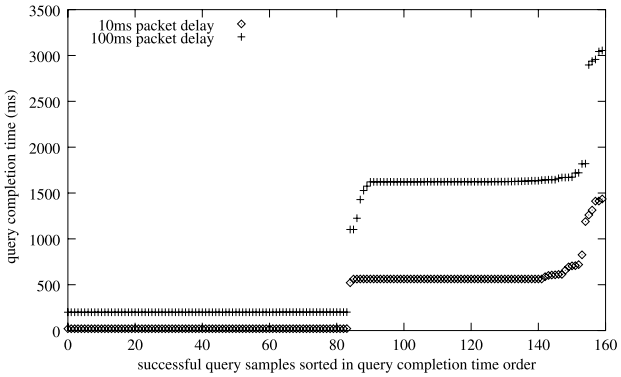

Fig. 12 Query completion time of DNS queries on 10 ms- and 100 ms-delay links with 1% packet loss rate while incoming DF-bit-set packets are blocked at the original DNS cache.

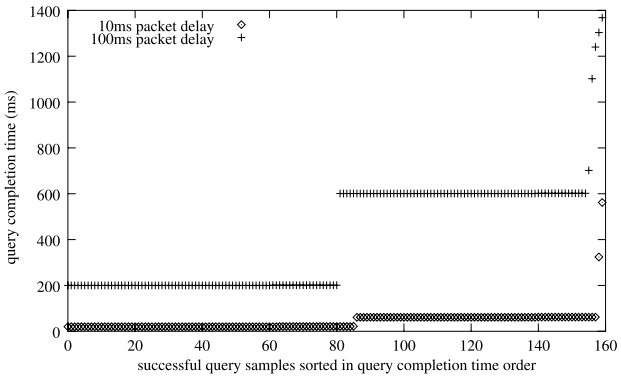

Fig. 13 Query completion time of DNS queries on 10 ms- and 100 ms-delay links with 1% packet loss rate while incoming DF-bit-set packets are blocked at the modified DNS cache.

5.4.3 Cases for Different Delay Time Values

We compare the test results of the cases of different delay times, for the 1% packet loss rate, when IP fragments are blocked. Table 7 shows the statistics of the queries. Figure 12 shows the results of the cases on the original cache, and the Fig. 13 shows those on the modified cache.

In the cases of 100 ms delay time, the typical query completion time for the original cache jumps from $\simeq$ 200 ms

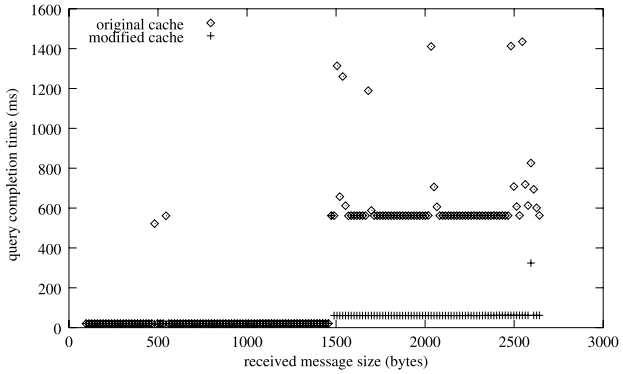

Fig. 14 Query completion time of DNS queries on 10 ms-delay link with 1% packet error rate while incoming DF-bit-set packets are blocked on DNS cache.

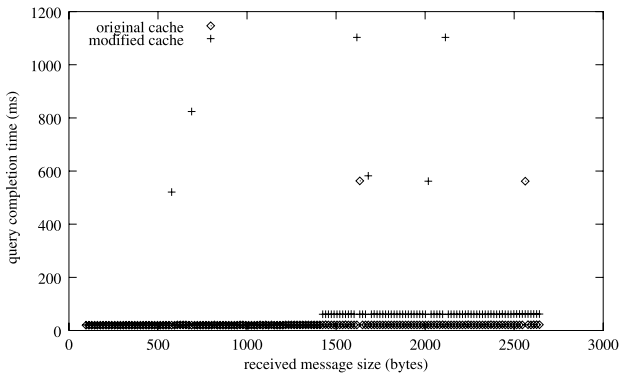

Fig. 15 Query completion time of DNS queries on 10 ms-delay link with 1% packet error rate while all IP fragments are allowed.

to $\simeq$ 1600 ms when the fragmentation started by the large answer message size. This is an effect of exponential timer back-off in BIND 9.4.2 resolver code, which adjusts the interval at `fctx_setretryinterval()` in `lib/dns/resolver.c`. The back-off algorithm is set to retry every 500 ms the first two times, and then back-off exponentially. On the other hand, the query completion time for the modified cache only jumps from $\simeq$ 200 ms to $\simeq$ 600 ms, which is due to the UDP-to-TCP fallback sequence when the EDNS0 UDP size value is set lower by our detection scheme.

In either delay time, the modified cache gives shorter mean values of query completion time.

5.4.4 Probing Characteristics of Our Proposed Scheme

Figure 14 shows the query completion time on 10 ms delay time and 1% packet error when IP fragments are blocked. This figure shows that the query response time will increase by the packet loss error, especially when the answer message size is large and requires TCP fallback.

During the test, we notify detection failures of our proposed scheme. Fig. 15 shows the query completion time on 10 ms delay time and 1% packet error when IP fragments are allowed. If no error occurs, the modified cache would use UDP query for all the tried message sizes. Fig. 15 shows instead that the modified cache use TCP for the large-size queries, as directed in the pseudo-code (Fig. 7).

Our modification remembers the change of the timeout status for the fragmentation detection, during the sequential testing of querying different A RRs. In our proposed scheme, there is no way to distinguish the reason of a timeout, whether it is caused by a packet error loss or a blockage of IP fragments. We conclude that the evaluated code may erroneously detect the IP fragmentation blockage due to the packet error.

5.5 Summary and Discussion of the Evaluation

In this section, we proposed our scheme and the implementation as a modification of BIND 9.4.2 resolver code. We evaluate the modified code as a DNS cache, comparing to the original (unmodified) cache. The results suggest the following characteristics of our proposed scheme:

- Our proposed scheme will reduce the query completion time when IP fragments are blocked between the end-to-end link of a DNS resolver and a server, especially on high-error-rate and high-latency links.
- In a very-low-error-rate link such as 100BASE-TX Ethernet, our proposed scheme does not impose large overhead even when IP fragments are allowed, by remembering the fragmentation allowance status for each destination host.
- Our proposed scheme may erroneously detect the packet loss as the IP fragment blockage. To prevent the adverse effect of this detection, periodical probing of the IP fragmentation status should be conducted.

We should also notice the common behavior of authoritative servers of BIND and NSD during the evaluation test. In either software package, the DNS server code directs the resolver to fall back to TCP by setting the TC (truncation) bit of the DNS answer, when the server has to send a larger message than specified by the resolver's EDNS0 UDP size value. This behavior has not been explicitly documented in EDNS0 specification.

6. Conclusion

This paper discusses a scheme to detect the UDP large-payload transfer capability between two DNSSEC hosts over the DNS application protocol. Current DNS protocol lack the active detection of maximum application MTU and causes the performance degradation. We propose a scheme of sending a large UDP query, which is EDNS0-compliant and can change the payload length while maintaining the conformance to the DNS specifications. We have shown the feasibility of the propose scheme by testing it over multiple DNS programs and confirmed the consistency of the tested results by the source code analysis. We have also proposed an implementation of the proposed scheme for shortening the transition time on fragment-blocked transports, and have

conducted the evaluation of the implementation. The evaluation results show our proposed scheme decreases the query completion time over fragment-blocked transports.

The future work include the security implication analysis of allowing large-payload UDP datagrams, optimization algorithms of DNSSEC server answer length for a given MTU value, and UDP retransmission strategy analysis for the optimal use of the network bandwidth.

Acknowledgment

This work is supported by NICT Incentive Research Fund for FY2006.

References

[1] P.V. Mockapetris, "Domain names – Concepts and facilities," RFC1034 (also STD13), 1987.
[2] P.V. Mockapetris, "Domain names – Implementation and specification," RFC1035 (also STD13), 1987.
[3] R. Elz and R. Bush, "Clarification to the DNS specification," RFC2181, 1997.
[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," RFC4033, March 2005.
[5] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Resource records for the DNS security extensions," RFC4034, March 2005.
[6] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Protocol modifications for the DNS security extensions," RFC4035, March 2005.
[7] P. Vixie, "Extension mechanisms for DNS (EDNS0)," RFC2671, 1999.
[8] O. Gudmundsson, "DNSSEC and IPv6 A6 aware server/resolver message size requirements," RFC3226, Dec. 2001.
[9] R. Braden (Editor), "Requirements for Internet hosts – Communication layers," RFC1122, 1989.
[10] S. Deering and R. Hinden, "Internet protocol, version 6 (IPv6) specification," RFC2460, 1998.
[11] C.A. Kent and J.C. Mogul, "Fragmentation considered harmful," 1987. Digital Equipment Corporation Western Research Laboratory Technical Report 87.3, http://www.hpl.hp.com/techreports/Compaq-DEC/WRL-87-3.html
[12] G. Ziemba, D. Reed, and P. Traina, "Security considerations for IP fragment filtering," RFC1858, 1995.
[13] I. Miller, "Protection against a variant of the tiny fragment attack," RFC3128, 2001.
[14] K. Rikitake, K. Nakao, S. Shimojo, and H. Nogawa, "A study of DNSSEC operation and deployment," IEICE Technical Report, ICSS2006-06, April 2006.
[15] Internet Corporation for Assigned Names and Numbers (ICANN) Security and Stability Advisory Committee (SSAC), "Testing firewalls for IPv6 and EDNS0 support," SSAC Report SAC 016, Jan. 2007. http://www.icann.org/committees/security/sac016.htm
[16] R. Braden, ed., "Requirements for Internet hosts – Application and support," RFC1123, 1989.
[17] S. Thomson, C. Huitema, V. Ksinant, and M. Souissi, "DNS extensions to support IP version 6," RFC3596, 2003.
[18] K. Rikitake, H. Nogawa, T. Tanaka, K. Nakao, and S. Shimojo, "DNS transport size issues in IPv6 environment," Proc. 2004 International Symposium of Applications and the Internet (SAINT2004) Workshops, pp.141–145, 2004.
[19] K. Rikitake, H. Nogawa, T. Tanaka, K. Nakao, and S. Shimojo, "An analysis of DNS payload length increase during transition to IPv6," IEICE Trans. Commun. (Japanese Edition), vol.J87-B, no.10, pp.1552–1563, Oct. 2004.
[20] K. Rikitake, A Study of DNS Transport Protocol for Improving The Reliability, Ph.D. Dissertation, Graduate School of Information Science and Technology, Osaka University, Osaka, Japan, Dec. 2004. Public Release Version 1.0, http://www.ne.jp/asahi/bdx/info/depot/dnstransport-phd-v10-pub.pdf
[21] D. Eastlake, "RSA/SHA-1 SIGs and RSA KEYs in the domain name system (DNS)," RFC3110, May 2001.
[22] B. Ager, H. Dreger, and A. Feldmann, "Exploring the overhead of DNSSEC," April 2005, http://www.net.informatik.tu-muenchen.de/~anja/feldmann/papers/dnssec05.pdf
[23] K. Rikitake, H. Nogawa, T. Tanaka, K. Nakao, and S. Shimojo, "An analysis of DNSSEC transport overhead increase," IPSJ SIG Technical Reports 2005-CSEC-28, vol.2005, no.33, pp.345–350, March 2005.
[24] E. Allman, J. Callas, M. Delany, M. Libbey, J. Fenton, and M. Thomas, "DomainKeys identified mail (DKIM) signatures," RFC4871, May 2007.
[25] M. Wong and W. Schlitt, "Sender policy framework (SPF) for authorizing use of domains in e-mail, version 1," RFC4408, April 2006.
[26] J. Klensin, ed., "Simple mail transfer protocol," RFC2821, April 2001.
[27] J. Schlyter and W. Griffin, "Using DNS to securely publish secure shell (SSH) key fingerprints," RFC4255, Jan. 2006.
[28] T. Ylonen and C. Lonvick, ed., "The secure shell (SSH) protocol architecture," RFC4251, Jan. 2006.
[29] M. Richardson, "A method for storing IPsec keying material in DNS," RFC4025, March 2005.
[30] S. Kent and R. Atkinson, "Security architecture for the internet protocol," RFC2401, 1998.
[31] S. Josefsson, "Storing certificates in the domain name system (DNS)," RFC4398, March 2006.
[32] A. Gulbrandsen, P. Vixie, and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)," RFC2782, Feb. 2000.
[33] M. Mealling, "Dynamic delegation discovery system (DDDS) part one: The comprehensive DDDS," RFC3401, Oct. 2002.
[34] M. Mealling, "Dynamic delegation discovery system (DDDS) part two: The algorithm," RFC3402, Oct. 2002.
[35] M. Mealling, "Dynamic delegation discovery system (DDDS) part three: The domain name system (DNS) database," RFC3403, Oct. 2002.
[36] M. Mealling, "Dynamic delegation discovery system (DDDS) part four: The uniform resource identifiers (URI) resolution application," RFC3404, Oct. 2002.
[37] P. Faltstrom, "E.164 number and DNS," RFC2916, Sept. 2000.
[38] M. Stapp, T. Lemon, and A. Gustafsson, "A DNS resource record (RR) for encoding dynamic host configuration protocol (DHCP) information (DHCID RR)," RFC4701, Oct. 2006.
[39] R. Droms, "Dynamic host configuration protocol," RFC2131, 1997.
[40] R. Droms, ed., J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic host configuration protocol for IPv6 (DHCPv6)," RFC3315, 2003.
[41] M. Stapp, B. Volz, and Y. Rekhter, "The dynamic host configuration protocol (DHCP) client fully qualified domain name (FQDN) option," RFC4702, Oct. 2006.
[42] M. Stapp and B. Volz, "Resolution of fully qualified domain name (FQDN) conflicts among dynamic host configuration protocol (DHCP) clients," RFC4703, Oct. 2006.
[43] B. Volz, "The dynamic host configuration protocol (DHCP) client fully qualified domain name (FQDN) option," RFC4704, Oct. 2006.
[44] P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, "Dynamic updates in the domain name system (DNS UPDATE)," RFC2136, 1997.
[45] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session initiation protocol," RFC3261, June 2002.
[46] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time applications," RFC3550, July 2003.
[47] K. Rikitake, K. Nakao, S. Shimojo, and H. Nogawa, "IP fragmenta-

tion and the implication in DNSSEC," IPSJ SIG Technical Reports, 2007-CSEC-37, vol.2007, no.48, pp.79–84, May 2007.
[48] The FreeBSD Project, "FreeBSD," http://www.freebsd.org/
[49] J. Mogul and S. Deering, "Path MTU discovery," RFC1191, 1990.
[50] J. McCann, S. Deering, and J. Mogul, "Path MTU discovery for IP version 6," RFC1981, 1996.
[51] D. Mills, "Simple Network Time protocol (SNTP) version 4 for IPv4, IPv6 and OSI," RFC2030, 1996.
[52] NIC-SE, "`.se` is the first TLD in the world with DNSSEC – A more secure technique for name resolving on the Internet," NIC-SE Press Release on Sept. 14, 2005, http://www.nic.se/english/nyheter/pr/2005-09-14?lang=en
[53] Internet Software Consortium, "BIND," http://www.isc.org/bind/
[54] NLnet Labs, "Name server daemon (NSD)," http://www.nlnetlabs.nl/nsd/
[55] D.J. Bernstein, "djbdns," http://cr.yp.to/djbdns.html
[56] Internet Assigned Numbers Authority (IANA), "Domain name system parameters," http://www.iana.org/assignments/dns-parameters
[57] R. Austein, "DNS name servier identifier option (NSID)," June 2006. INTERNET-DRAFT draft-ietf-dnsext-nsid-02.txt.
[58] R. Vaughn and G. Evron, "DNS amplification attacks (*Preliminary Release*)," Dated March 17, 2006, http://www.isotf.org/news/DNS-Amplification-Attacks.pdf
[59] D. Conrad, "Indicating resolver support of DNSSEC," RFC3225, Dec. 2001.
[60] O. Kolkman, "Net::DNS," http://www.net-dns.org/
[61] L. Wall, "Perl," http://www.perl.org/
[62] D. Reed, "IP filter," http://coombs.anu.edu.au/~avalon/
[63] K. Rikitake, "A valid NSID EDNS0 option generates FORMERR on nsd-3.0.5," NLnet Labs Bugzilla Bug 157, http://www.nlnetlabs.nl/bugs/show_bug.cgi?id=157
[64] M. Mathis and J. Heffner, "Packetization layer path MTU discovery," RFC4821, March 2007.
[65] Common Vulnerabilities and Exposures (CVE), "CVE-2002-1220," http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2002-1220
[66] L. Mamakos, K. Lidl, J. Evarts, D. Carrel, D. Simone, and R. Wheeler, "A method for transmitting PPP over Ethernet (PPPoE)," RFC2516, Feb. 1999.
[67] L. Rizzo, "Dummynet: A simple approach to the evaluation of network protocols," Comput. Commun. Rev., vol.27, no.1, pp.31–41, 1997.
[68] TCPDUMP Public repository, "tcpdump," http://www.tcpdump.org/

Kenji Rikitake received the B.E. in instrumentation physics and M.E. in information engineering from University of Tokyo in 1988 and 1990, respectively. He has been a Japanese government-licensed professional engineer (P.E.Jp., Gijyutsushi) of information engineering since 2001. He received the Best Paper Award of the IPSJ 63rd National Convention in 2002. He received Ph.D. in information science from Osaka University in 2005. Since 2005, he has been an Expert Researcher of Network Security Incident Response Group of National Institute of Information and Communications Technology. His research interests include DNS, computer security, and teleworking. He is a member of IPSJ, ACM and IPEJ.

Koji Nakao received the B.S. degree of Mathematics from Waseda University in 1979. Since 2004, he has been the Group Leader of Network Security Incident Response Group of National Institute of Information and Communications Technology. Since 2007, he has been an Information Security Fellow of KDDI Corporation. He is a member of IPSJ.

Shinji Shimojo received the M.E and Dr.Eng degrees from Osaka University, in 1983 and 1986, respectively. Since 1998, he has been a Professor at the Cybermedia Center. His research interest includes distributed operating systems. He is a member of IPSJ, IEEE and ACM.

Hiroki Nogawa received M.D. and Ph.D. degrees in Medicine from Osaka University in 1990 and 1997, respectively. Since 2004, he has been a visiting professor at Tokyo Medical and Dental University. His research interest includes computer security, security management and public policy issues. He is a member of IPSJ.