A Study of DNS Transport Protocol for Improving the Reliability

（信頼性向上のための DNS トランスポートプロトコルの研究）

December 2004

Kenji Rikitake

力武　健次

A dissertation
submitted to the Graduate School
of Information Science and Technology
of Osaka University
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Information Science

Summary

The goals of this dissertation are to show how critical the negative effects imposed by the current 512-byte limitation of UDP payload length in DNS (Domain Name System) transport protocol are, and to propose solutions to work around the limitations in DNS operation while minimizing the migration cost by using the existing protocol enhancements.

DNS is a distributed database of Internet, which binds the domain names to the actual resource pointers such as IP addresses of Web servers and the name of mail exchanging hosts of a domain. The reliability of DNS defines the reliability of the whole Internet, since many application services, such as electronic mail and Web, are dependent on the authenticity of domain names.

DNS has its own transport protocol. Most of the exchange between DNS resolvers (clients) and servers complete in a single transaction. If the length of a payload is equal or less than 512 bytes, UDP is used, but if it exceeds 512 bytes, the transfer is switched to TCP after the UDP transfer is tried (RFC1035 Section 4.2.1). While this protocol design guarantees the fundamental reliability of DNS, switching to TCP from UDP is redundant, and has become a performance bottleneck for the expansion of DNS functions.

Recently the functional demands for DNS are changing a lot, from the day when it was formally defined in 1987 in RFC1034 and RFC1035. The new functional enhancements include the support for the emergence and migration to IPv6, DNSSEC for cryptographic authentication of DNS, and the dynamic updates of DNS contents. These new enhancements have a common characteristic to increase the length of payloads for each exchange. The average payload length of DNS exchange is increasing as these new enhancements are gaining popularity.

As the average payload length of DNS exchange increases, the percentage of DNS exchange over TCP which carries the payloads larger than 512 bytes in the whole DNS exchanges also gets larger. This increase causes the larger usage of the processing power and computational resource of DNS servers and caches. The switching to TCP also contributes to raise the number of packets exchanged between the resolvers and servers, which results in more network bandwidth consumption and higher packet-exchange performance demand for the network to which large-scale DNS servers are connected, and less efficiency and reliability of the whole

DNS.

Some protocol enhancements to overcome the 512-byte limitation of DNS exchange over UDP are proposed. EDNS0 in RFC2671 defines an extension to indicate for a larger payload length of each UDP exchange of DNS. T/TCP (Transactional TCP) in RFC1644 defines an extension to TCP itself to reduce the number of packets for a transactional (single round-trip exchange) use of TCP to reduce the numbers of packets while maintaining the reliability of TCP. Many DNS operators claim the early deployment of these enhancements is important to prevent the negative effect of the UDP payload-length limitation.

In this dissertation, the author first discusses the historical role of DNS, DNS operation and Internet security, and the current issues in Chapter 1 as an introduction. The author also explains the DNS architecture and the transport protocol in details in Chapter 2. The layering of DNS protocols is explained, and the characteristics and functions of each protocol layer are discussed, as well as the detailed transport protocol specification using existing implementations. The author also shows emerging functional demands to the DNS transport protocol as newly-emerged enhancements are gaining popularity, and proposes the possible solutions.

The author analyzes the effect of payload-length increase of DNS in Chapter 3, using the migration of IPv4 to IPv6 as an example, with the real-world traffic data and a simulation which reflects the IP-address length increase from IPv4 to IPv6. The introduction of IPv6 changes the length and contents of DNS traffic due to the increased address length and other enhancements. The author reviews how DNS protocols are affected by the IPv6 transition, and shows the percentage of DNS answers exceeding the 512-byte UDP payload size limit, including the additional records, increases from 0.04% to 1~3% with a simulation by packet-length recalculation, using the real-world DNS traffic. The author then shows the quantitative effectiveness of EDNS0 to increase the acceptable payload size.

The author then proposes introducing T/TCP to reduce the overhead of TCP-based DNS exchange to reduce the impact of current 512-byte limitation of UDP-based DNS exchange in Chapter 4. The author evaluates the T/TCP by implementing the protocol to existing DNS program codes and measuring the increase of efficiency using real-world ADSL Internet link and a simulated link for a mobile Internet access. The author then shows a conclusion that T/TCP is an effective alternative to improve the efficiency of DNS exchange, especially for mobile Internet access with a minimal development overhead of protocol development.

Finally in Chapter 5, the author concludes this dissertation and discusses the direction of future works, including how the DNS should evolve and the functions of transport protocol required to maintain the reliability of DNS.

Contents

Chapter 1

Introduction

1.1 The Historical Role of DNS and the Current Issues

One of the most important and critical subsystems of the Internet Protocol Suite is DNS (Domain Name System). Many mission-critical applications depend on DNS for the domain name resolution. For example, Electronic mail messages use domain names to choose the source and destination addresses. The Web is fully dependent on the integrity of domain names to specify the appropriate servers.

DNS has been a mandatory component of the Internet since the establishment in 1983 by Mockapetris [1] and other Internet researchers and engineers. The main purpose of DNS was to replace the `HOSTS.TXT`, a single-file text database which contained the mapping of the whole Internet hosts and the corresponding IP (Internet Protocol) addresses.

Before DNS came into being, the host database of Internet was solely managed by updating the `HOSTS.TXT`. Maintaining the `HOSTS.TXT` required a centralized management and distribution, and as the scale of Internet grew up the management overhead became too high. Instead of depending to a file, DNS is established to maintain a collective set of distributed database to keep the integrity of domain name resolution, while the delegation of database maintenance authority is given to each connected entity, an organization or an individual.

The current specification of DNS is revised by Mockapetris as described in Internet RFCs (Request For Comments) RFC1034 [2] and RFC1035 [3], also supplemented by RFC1123 [4] Section 6, RFC2181 [5], and many other RFCs.

Each database for a Internet-connected entity contains at least one *Zone*, a set of RRs (Resource Records) which represents the hosts and other operation information of the domain representing the entity. The zones are linked with the hierarchical delegation from the Root Zone, with a set of top-down authorization of the domain name registries. The authorization path also represents the political bodies of the governance of the Internet.

Figure 1.1 shows a typical example of the zone hierarchy which begins with the Root

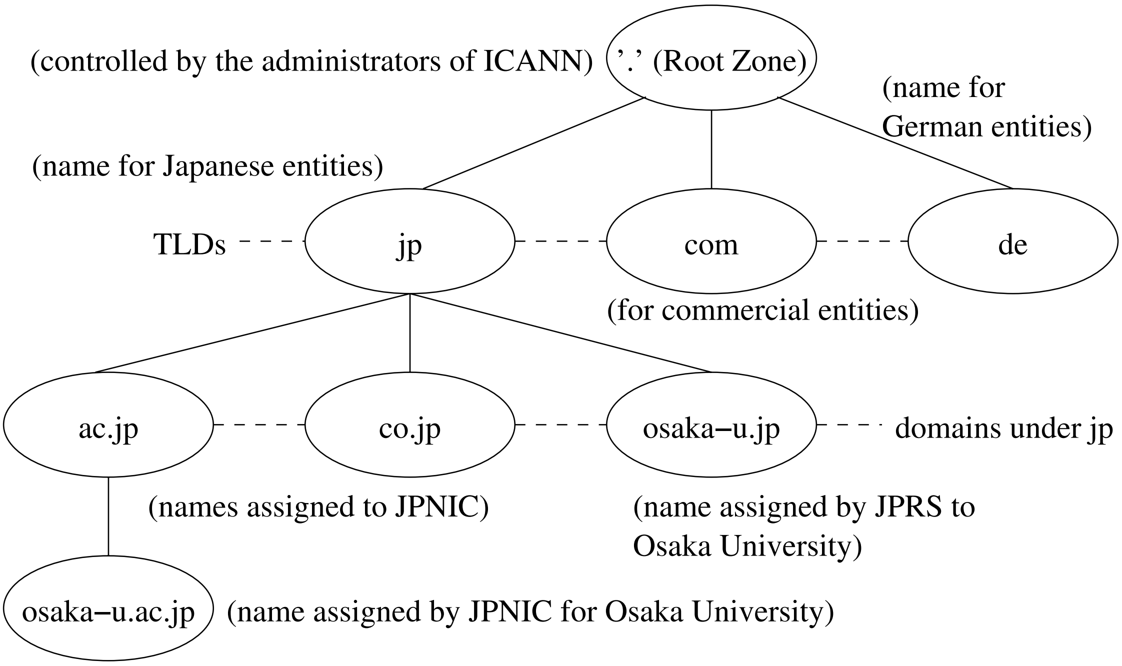

Figure 1.1 An example of domain name delegation tree

Zone. The Root Zone is under control of ICANN (Internet Corporation For Assigned Names and Numbers), who also has the ultimate control over the TLDs (Top-Level Domains). Two major categories exist in the TLDs: by country (separated by geographic regions) and by general attributes (such as `.com` for commercial entities, etc.).

ICANN delegates the authority of assigning domain names to the country-level registries such as JPNIC (Japan Network Information Center) and JPRS (Japan Registry Service), as well as to the other registries, many of which are run by private companies.

Each endpoint entities must register their names to the registry of the adjunct upper-level domain. The registration link also represents the delegation of authority for domain names. For example, to use `osaka-u.ac.jp`, the name must be registered to JPNIC, which controls the `ac.jp`, an attribute-based subdomain of `jp`. On the other hand, to use `osaka-u.jp`, the name must be registered to JPNIC through JPRS, which has the authority granted from JPNIC for assignments of non-attributal subdomains of `jp`.

Table 1.1 shows some examples of well-known DNS RR (Resource Record) types and how they are used. DNS is essential to provide fundamental resource pointers for Internet, including:

- bidirectional references between domain names and IP addresses;
- mail exchanger host for a domain, which acts as the mail receiver for the domain;

Table 1.1 Some examples of DNS RR types

A	IPv4 address assigned to a domain name (usually a host)
AAAA	IPv6 address assigned to a domain name (usually a host)
CNAME	Canonical name for an alias (used for specifying a service-based alias, such as `www.osaka-u.ac.jp`, for a host which has another real host name)
MX	Mail eXchanger host name for a domain
NS	Authoritative name server host name
OPT	Specifying options for EDNS0 [6], negotiating protocol extension capabilities
PTR	Domain name pointer for a IP-address-mapped domain name (used for reverse lookups)
SOA	Start of authority for a zone or a domain (contains the serial number used for updating the zone data, name server host name for the primary source of the zone data, contact mailbox, etc.)
TXT	Generic text data associated with the name (used by various applications)

- aliases for other hosts, which enables to assign multiple reference names for a host which provides multiple kinds of services;
- names of authoritative name servers, to show the proper path of domain namespace delegation of authority;
- extension of DNS protocol itself using EDNS0; and
- association of arbitrary text data to a domain name, which can be used by external application services.

New protocol extensions to DNS have been brought into the specification, regarding the recent changes of the Internet usage, including:

- support for IPv6 address and reverse-lookup references, including AAAA RRs and PTR RRs to `ip6.arpa` [7] domain, which is essential to address resolution and host name authentication;
- DNSSEC [8], to cryptographically authenticate RRs by attaching additional RRs of the digital signatures; and
- DNS UPDATE [9], to dynamically update the Zone database by end nodes, designed for reflecting the change of IP address in a mobile operation environment.

Those new extensions demand more secure and reliable data exchange between DNS resolvers and servers, as the length of payload for each transaction increases and the content of payload must not be altered in any way.

1.2 DNS Operation and Internet Security

DNS operation is a critical part of overall Internet security. The integrity and availability of Internet depends on how DNS, the zones, and the RRs are managed. Improperly managed DNS database often turns out to be a persistent problem of security, and difficult to properly fix for the network administrators.

For example, the address-to-domain-name resolution of IP (Internet Protocol) fully depends on DNS. The IPv4 (IP version 4) address resolution is performed by looking up the `in-addr.arpa` domain database. If entries of the database are incorrect, the whole integrity of this resolution method, is lost.

The availability of Internet also depends on the reliability of DNS contents and the data transport. If a DNS RR contains a wrong, incomplete or maliciously spoofed (impersonated or fabricated) content, or if a DNS transport protocol between the resolvers and servers is incorrectly or incompletely performed, the results are communication disruption by authentication failures, or even worse hijacked connections whose clients are misdirected to the attacker's servers.

Many Internet application services heavily depends on DNS. For example, delivery of electronic mail is dependent on DNS. MX records for a domain must point the correct servers which handles the delivery for the domain. Otherwise, messages to the domain would not be delivered, or would wrongly be delivered to another irrelevant host or be directed to an attacker who intends to eavesdrop the contents of the messages.

HTTP [10] communication also depends on DNS. The authenticity of HTTP servers using unencrypted plaintext communication depends solely on the authenticity of DNS RRs, which tells the IP address of the servers to the clients. If an attacker could provide a spoofed DNS RR for an HTTP server, access requests to the server are directed to the attacker's server. This leads to a conclusion that the phishing fraud of Internet could be much easily done if the DNS server for the HTTP server is succeeded.

Even cryptographically-secure protocols are also dependent on DNS. TLS (Transport Layer Security) defined by RFC2246 [11] and RFC3546 [12], a security enhancement using encryption and cryptographic authentication primarily designed for HTTP and other applications based on TCP [13], does not guarantee if the TLS client connects to the authenticated server by itself, and assumes that the authenticity of the server is guaranteed through DNS mapping. While TLS prevents the server spoofing with its own cryptographic authentication, the client still depends on DNS to reach the right server.

Theoretically speaking, to make DNS resistant to security attacks, the whole traffic must be cryptographically authenticated and no spoofing will occur in any way. Unfortunately, the current DNS is mostly not cryptographically authenticated, since most of the resolvers do not support the authentication enhancement such as DNSSEC. IPsec [14], the still-evolving host-level IP-layer encryption and authentication, is still not gaining majority support on the Internet, and will not necessarily guarantee the end-to-end security between DNS resolvers and servers, especially the communication are performed through DNS caches.

On the other hand, the current data transport protocol between DNS resolvers and servers will encounter severe performance and reliability drawback when DNS contents are enhanced and the length are getting longer, for meeting the recently-emerged requirements such as transition to IPv6 and introduction of DNSSEC and DNS UPDATE, due to the 512-byte limitation of UDP [15] payload length (on RFC1035 Section 4.2.1) and the requirement of protocol fallback to TCP.

One thing the author should emphasize is that research of DNS security is not necessarily conducted on practical basis. For example, the mainstream of current research activities conducted by the `dnsext` (DNS extension) working group of IETF (Internet Engineering Task Force) are heavily directed towards introducing the public-key cryptographic authentication called DNSSEC [8] into the data exchange between the DNS servers and resolvers. While DNSSEC may help DNS programs to authenticate the exchanged data, it has many practical issues to be solved such as that the trust delegation mechanism is claimed not to be robust enough for the real-world deployment [16], and that the design is still subject to major changes [17].

Moreover, DNSSEC does not solve the major problems which DNS administrators currently face, since they are mostly based on non-cryptographic issues, such as the transport layer security to retain tolerance against DoS (Denial-of-Service) attacks while maintaining the availability of the service, the database content management, and the program code integrity of the servers and resolvers. Without a solid non-cryptographic data transport, a securely-authenticated link cannot be established. This means DNSSEC will not work without a non-cryptographic transport which is capable to transfer larger DNS payloads of DNSSEC signature RRs. In other words, DNS security would only be realized if the reliability and optimal performance of the transport are established. The research target in this dissertation is to make a reliable DNS transport as a foundation for upper-layer protocols, such as DNSSEC.

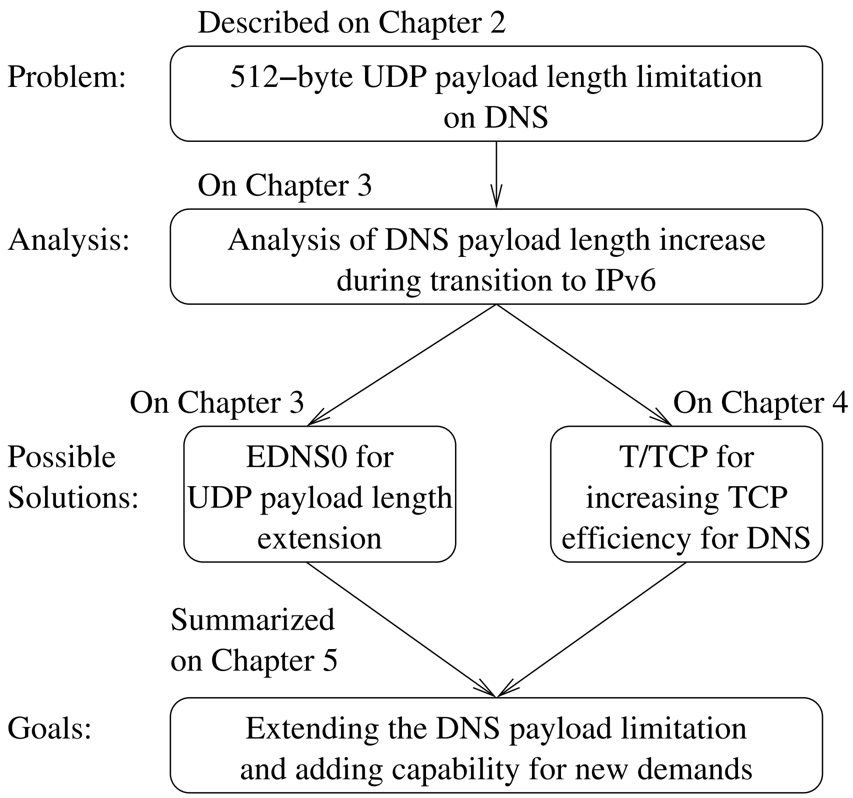

Figure 1.2 Outline flow of the dissertation

1.3 Outline of the Dissertation

The goals of this dissertation are to show how critical the negative effects imposed by the current 512-byte limitation of UDP payload length in DNS (Domain Name System) transport protocol are, and to propose solutions to work around the limitations in DNS operation while minimizing the migration cost by using the existing protocol enhancements.

The remainder of this dissertation is organized as follows (Figure 1.2). The author explains the DNS architecture and the transport protocol in details in Chapter 2. The layering of DNS protocols is explained, and the characteristics and functions of each protocol layer are discussed, as well as the detailed transport protocol specification using existing implementations. The author also shows current requirements to the DNS transport protocol as newly-emerged enhancements are gaining popularity, and proposes the possible solutions.

The author analyzes the effect of payload-length increase of DNS in Chapter 3, using the migration of IPv4 to IPv6 as an example, with the real-world traffic data and a simulation which reflects the IP-address length increase from IPv4 to IPv6. The introduction of IPv6 changes the length and contents of DNS traffic due to the increased address length and other enhancements. The author reviews how DNS protocols are affected by the IPv6 transition, and shows the percentage of DNS answers exceeding the 512-byte UDP payload size limit, including the additional records, increases from 0.04% to 1~3% with a simulation by packet-length recalculation, using the real-world DNS traffic. The author then shows the quantitative effectiveness of EDNS0 to increase the acceptable payload size.

The author then proposes introducing T/TCP to reduce the overhead of TCP-based DNS exchange to reduce the impact of current 512-byte limitation of UDP-based DNS exchange in Chapter 4. The author evaluates the T/TCP by implementing the protocol to existing DNS program codes and measuring the increase of efficiency using real-world ADSL Internet link and a simulated link for a mobile Internet access. The author then shows a conclusion that T/TCP is an effective alternative to improve the efficiency of DNS exchange, especially for mobile Internet access with a minimal development overhead of protocol development.

Finally in Chapter 5, the author concludes this dissertation and discusses the direction of future works, including how the DNS should evolve and newly-required functions of transport protocol to maintain the reliability of DNS.

Chapter 2

DNS Architecture and the Transport Protocol

2.1 Introduction

DNS works with a set of protocols; there is no single DNS protocol, and many different kinds of protocols involve in the DNS operation. DNS consists of multiple communication layers, and the protocols are closely related and dependent on each other. The official documents of DNS [2, 3] do not rigidly define the detailed behavior of each DNS programs either, and many important parts of DNS are slightly but surely different between different implementations.

Practically speaking, DNS evolves as the *de facto* implementation development proceeds. BIND (Berkeley Internet Name Daemon) [18] and the configuration files practically defines and limits many aspects of DNS. BIND has been a testbed for various DNS extensions, including DNSSEC and DNS UPDATE.

In this chapter, however, the author describes DNS using implementation-dependent explanation methods as much as possible, other than the implementation-specific issues, since quite a few other DNS servers such as `djbdns` [19] and NSD [20] have been emerging as production-level alternatives to BIND. The author also emphasizes on explaining the set of DNS data transport-related protocols for showing the effects of UDP 512-byte payload length limitation, and showing how current DNS issues to be analyzed and evaluated in the later sections.

In later sections, the author describes the protocol layers of DNS and the transport specification in Section 2.2, and shows a definition of the UDP 512-byte payload length limitation. The author then focuses on emerging new issues and how they contribute to make the payload length larger, such as supporting migration from IPv4 to IPv6 in Section 2.3, the RR authentication and DNSSEC in Section 2.4, and the dynamic update of DNS contents in Section 2.5. The author concludes this chapter in Section 2.6.

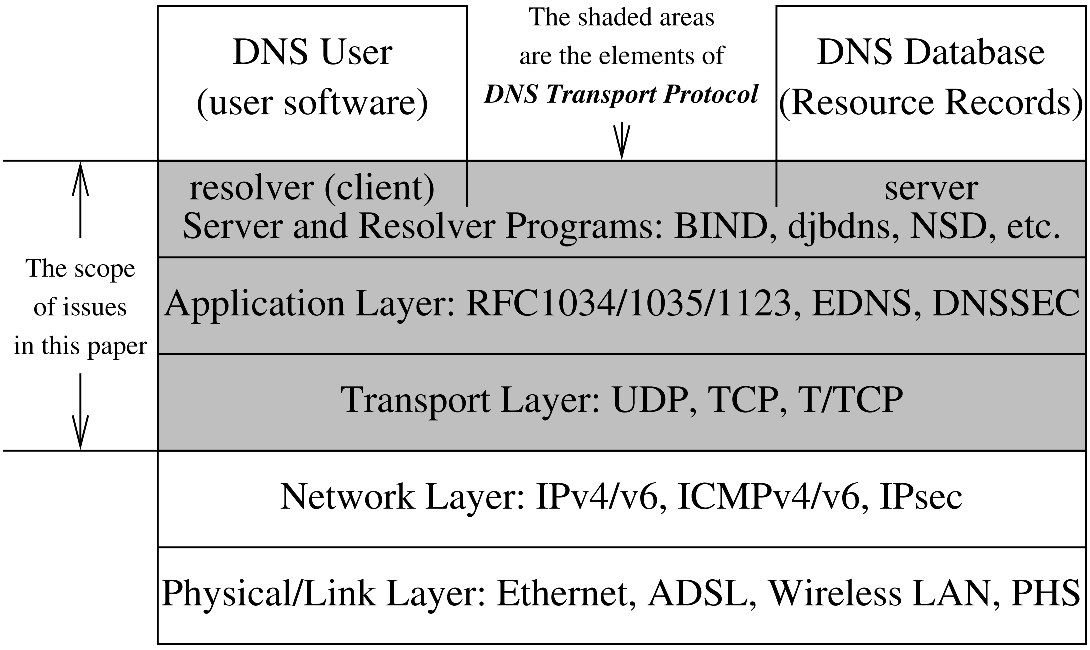

Figure 2.1 DNS protocol layers and the elements

2.2 DNS Protocol Layers and Transport Specification

In this section, the author describes the layers of DNS components and the protocols.

Figure 2.1 shows a simplified model of DNS component programs and protocols, described as a stack of multiple layers. In this model, DNS provides a mechanism for the users including other application programs to retrieve and update the database entries, or RR (Resource Records), located on the servers.

In this dissertation, the author defines the *DNS Transport Protocol* as a collective set of protocols, of the shaded part of Figure 2.1, as follows:

- the TCP/IP transport layer protocols, including UDP and TCP;
- DNS application layer protocols, defines in RFC1034, RFC1035 and other extensions; and
- implementation-specific behavior of actual servers and resolvers, including those of BIND and `djbdns`.

DNS Transport Protocol defines the characteristics of DNS data transport between the servers and resolvers, by the characteristics and behaviors of each protocol layer and element, and the interaction between the elements and layers.

2.2.1 DNS Protocol Layers

The following is a list of description for each layer in Figure 2.1, from the bottom to the top:

2.2.1.1 Physical/link Layer

This layer consists of the physical media and the link-layer protocol of data exchange. The upper layers should be able to reliably handle and to equally treat the physical media and the link layer, which may have different latencies, bandwidths, and packet loss rates. These layers should guarantee adequate reliability of data transfer required by the upper layers.

In this dissertation, the author assumes that the physical and link layers guarantee lower enough packet loss rate and latency, and higher enough bandwidth to perform practical communication between DNS servers and resolvers, so that the communication conditions of these layers do not cause significant problems to the upper-layer communication.

2.2.1.2 Network Layer

This layer consists of the IP (Internet Protocol, both version 4 and version 6) and the related protocol components, such as ICMP (Internet Control Message Protocol). In this layer the upper-layer data could be split into fragment packets and reassembled, to keep the packet size lower than the limit imposed by the physical/link layer.

One of the recent development on this layer is IP Security or IPsec [14], the cryptographic authentication and data encryption capability of IP, is an important optional component. IPsec is widely used for protecting IP packets from spoofing and monitoring, though it does not guarantee the upper-layer data integrity, since it only protects host-to-host IP-level packets.

On DNS, the authentication feature of IPsec will largely contribute to ensure the data integrity of DNS payloads and RRs, while the encryption feature of IPsec is not necessarily required.

2.2.1.3 Transport Layer

This layer traditionally consists of two protocols: TCP (Transmission Control Protocol) [13] and UDP (User Datagram Protocol). T/TCP [21] is a TCP enhancement for transactional data exchange. UDP provides the functionality of selecting data flow between different application services by assigning *port numbers* to each service, as well as the per-packet checksum to ensure the data integrity of each packet. TCP adds the retransmission functionality to provide a reliable communication between the application programs under data errors and packet losses. UDP has no notion of connection, while TCP has. UDP has less control on the packet filters and the proxy servers used at firewall devices. Reliability of this layer significantly affects the overall security of DNS.

DNS programs must make the UDP service port open to the external hosts to communicate with other programs. A host-level end-to-end latency-measurement tool [22] uses this wide availability of DNS service to estimate the latency by getting access to nearby DNS servers.

The openness of the UDP service port for DNS makes the host which the DNS programs are running prone and vulnerable to external UDP-based DoS attacks, since all UDP packets must be directly handled by the DNS programs. DoS attacks to UDP ports are much easier than those to TCP ports because the attackers do not have to maintain the connection states.

Recently some new transport protocols are developed. One of the notable protocols is SCTP (Stream Control Transmission Protocol) [23, 24], which provides higher resistance against flooding and masquerading DoS attacks by exchanging cookies before the actual communication starts. An SCTP-based DNS exchange might be useful to prevent DDoS (Distributed DoS) attacks.

On the other hand, some useful protocol extensions developed in a long time ago are rediscovered and gaining popularity. TCP MD5 Checksum Option [25], published in 1992, was not paid much attention until BGP (Border Gateway Protocol) [26], a core wide-area routing protocol of Internet, had to be secured from widely-known attacks to TCP such as the Sequence Number Attacks [27]. Cryptographically authenticating TCP packets is a good alternative to prevent unwanted forgery of TCP-based data exchange between well-known large-scale DNS servers and resolvers.

2.2.1.4 Application Layer

DNS server-resolver communication protocol is collectively defined by many Internet RFCs (Request For Comments). The two important RFCs are RFC1034 [2], which specifies the architecture of DNS, and RFC1035 [3], which specifies the implementation details of DNS. RFC1123 [4] also specifies the DNS usage as a part of the Internet host requirements. Some clarification based on the practical expertise is also given on a later document RFC2181 [5].

Some of the recent research proposals including the protocol extension frameworks called EDNS0 [6], DNSSEC, and the secure dynamic data update extension [28], have already been put into the production-level DNS implementations. These proposals are designed for meeting the demands of handling newly-emerged needs of IPv6, DNS authentication, and making DNS from a near-read-only to read-and-write distributed database.

Most of the DNS programs especially those running in production-level computer systems, however, are *still not capable* of performing those under-development extensions of DNS.

So in the production-level systems, the overall security and reliability of DNS should yet be considered *without the extensions*, including DNSSEC, as of December 2004. The author discusses the specific issues on DNSSEC later in Section 2.4.1.

2.2.2 Server and Resolver Programs

DNS program packages belong here. A DNS program package has its own resolver library which provides programming interfaces to lookup DNS database, the database lookup programs for administrative use, the cache programs for optimizing outbound DNS traffics, and the server programs for providing the DNS database information.

2.2.2.1 BIND: The Reference Implementation

As of December 2004, the most popular package is BIND [18], which is bound to many operating system distributions. The latest *stable* releases are versions 4.9.11, 8.4.5, 9.2.4, though the development release is version 9.3.0, as of December 2004. Use of older version are not recommended, though many production system hosts are still using the version-8-based DNS software and some even use the version-4-based software. BIND 9.3.0 has the latest implementations of various extensions, including DNSSEC, DNS UPDATE, and EDNS0.

BIND, which has been a part of UNIX operating systems derived from BSD (Berkeley Standard Distribution), has been the *de facto* standard of DNS implementation since global DNS operation has begun in late 1980s, for many reasons as follows:

- Many of DNS servers have been, and still are, run on BSD-derived UNIX machines, including FreeBSD [29], OpenBSD [30] and NetBSD [31];
- BIND is ported to many architectures, including popular non-BSD UNIXes such as Linux [32] distributions;
- Most of existing programs solely use BIND's resolver library as it is the operating system standard; and
- Many Internet documents including RFCs use notations of BIND configuration files and zone files for explaining the technical issues.

As a result of its overwhelming popularity as a DNS program, BIND has fallen victim to various types of successful stack-smashing (buffer-overflow) and DoS attacks. For example, a set of bugs on the BIND resolver library [33, 34, 35] forced major OS distributions such as FreeBSD to upgrade [36]. Another set of bugs expose vulnerabilities of BIND DNS server which allows to execute an arbitrary code or to crash the server program and/or the host operating system [37, 38, 39]. Some versions of BIND has been discovered to be prone to DoS

```
$ORIGIN example.org.
@         42m40s IN SOA   dns0 hostmaster (
                          1094215095      ; serial
                          4h33m4s         ; refresh
                          34m8s           ; retry
                          1w5d3h16m16s    ; expiry
                          42m40s )                ; minimum
          3D IN NS        dns0
dns0      3D IN A         xxx.xxx.xxx.xx4
@         3D IN NS        dns1
dns1      3D IN A         xxx.xxx.xxx.xx5
          1D IN MX        0 mx
mx        1D IN A         xxx.xxx.xxx.xx4
          1D IN MX        0 mx
dns2      1D IN A         xxx.xxx.xxx.xx6
          1D IN MX        0 mx
dynhost   3D IN NS        dns2
fixedhost 1D IN A         xxx.xxx.xxx.xx7
```

Figure 2.2 An example of BIND zone configuration file (output of `nslookup` program of BIND Version 8)

attacks [40]. Regarding this vulnerability history, the author may suspect that BIND has a serious problem on the software development, if not on the quality of the code itself.

2.2.2.2 `djbdns`: A Secure Alternative

A package called `djbdns` [19] is popular for production-level systems whose operational security is critical. While BIND uses the all-in-one approach for designing the server, `djbdns` uses a modular approach of system design, providing different executable programs for different purposes. This modularity makes `djbdns` highly configurable and managable for system administrators. For example, `djbdns` separates the cache for resolvers as the program `dnscache` and the authoritative (non-recursive) server program `tinydns`, while most of BIND name servers `named` contain both the cache and authoritative server functionalities in the same program.

Another operating characteristics of `djbdns` is that it does not assume explicit zoning as

```
# example.org tinydns data
# line begins with .(dot) stands for
# an authoritative NS RR (and the SOA RR)
.example.org:xxx.xxx.xxx.xx4:dns0.example.org.
.example.org:xxx.xxx.xxx.xx5:dns1.example.org.
# line begins with @(atmark) stands for a MX RR
@example.org::mx.example.org.
# line begins with + stands for an A RR
+mx.example.org:xxx.xxx.xxx.xx4
@mx.example.org::mx.example.org.
+dns2.example.org:xxx.xxx.xxx.xx6
@dns2.example.org::mx.example.org.
# line begins with + stands for an NS RR
# for a delegated domain
&dynhost.example.org::dns2.example.org.
+fixedhost.example.org:xxx.xxx.xxx.xx7
# end of tinydns data
```

Figure 2.3 An example of `tinydns` database configuration file (the source of Figure 2.2)

in the BIND zone configuration files, but it rather serves a set of RRs pre-defined in a database, not restricted in the domain name literal syntax. Figures 2.2 and 2.3 show the difference of the configuration philosophy between the two software. The BIND's file in Figure 2.2 configures a zone `example.org` using the `$ORIGIN` directive, but the `tinydns`'s file in Figure 2.3 is rather a plain listing of necessary RRs in a zone.

Known specific vulnerabilities of `djbdns` has not been reported yet, as of December 2004. `djbdns` does not support DNSSEC or EDNS0 yet, as in the latest release `djbdns-1.05`.

2.2.2.3 NSD: An Authoritative-only Server

A package called NSD [20], an authoritative-only DNS server designed for a high performance use such as in a Root Server, is also designed and under production-system use in Europe. NSD supports DNSSEC and has declared the commitment to make it enable as the default configuration, once DNSSEC is standardized.

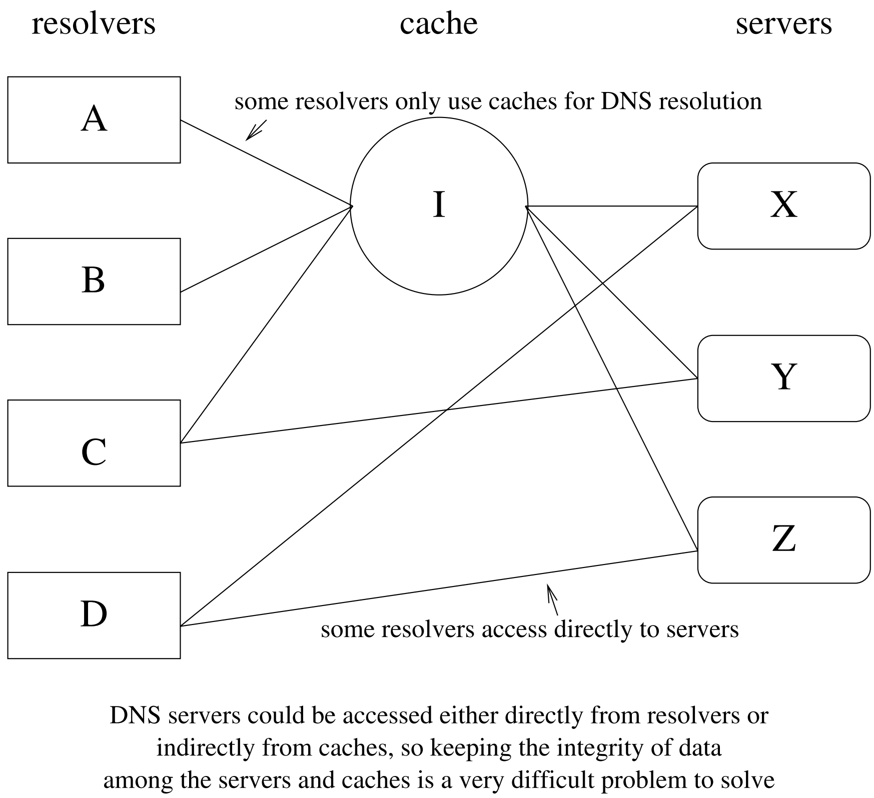

Figure 2.4 Variety of access methods from resolvers to servers with or without caches

2.2.3 Users and Databases

An entity connected to Internet with its own domain name must maintain the set of RRs of the domain under the DNS servers of its control. DNS has a distributed network of databases, as the servers form their network of delegation. Maintaining DNS database consistency among the servers is critical for minimizing the lookup overhead and preventing illegitimate RRs to be distributed.

Traditionally speaking, DNS contents were given statically by the zone administrator who has the responsibility of maintaining a mapping database such as that of between IP addresses and the host names, or mail exchanging host names to the hosts or subdomains within the zone. DNS authoritative servers, which serves the contents of DNS database which mostly consists of RRs, assumes that the database content updates are performed while the servers are *not* running. This static nature of DNS database allows loosely-coupled distributed servers and caches working simultaneously while providing the integrity of zone data, and allows the freedom of choice to resolvers for using or not using the DNS cache for the name resolution as shown in Figure 2.4.

While still most of the DNS databases are read-only and manually maintained by the administrator, allowing dynamic updates on the database is being utilized by the network sites which have dynamically-configured client hosts. DNS UPDATE [9] allows the update of DNS database contents by a transport protocol message. TSIG authentication [41], as well as DNSSEC, are used to authenticate the sender of the content update message, as well as using some non-cryptographic methods such as limiting the source IP address of the update message.

Vixie and Kato [42] describe that DNS can be used as a distributed dynamically-updateable database for a *real-time blackhole list* [43] to block unsolicited electronic mail messages, by combining the following extensions of DNS:

- DNS incremental zone transfer [44], which enables the zone transfer protocol of DNS to send only the differences between the latest and the older versions;
- DNS NOTIFY [45], a protocol between servers in the master-slave relationship, by which the master server advises the slave servers that the master's data has been changed and that the slaves should initiate the query for the database update; and
- DNS UPDATE, which enables the DNS clients to send the updated contents to the servers.

DNS dynamic update, however, should only be allowed with extreme care, since it may allow intruders to alter the DNS database contents. DNS also has a very complex reference architecture as shown in Figure 2.4, because the resolvers can make an arbitrary choice for using the caches. For example, in Figure 2.4 resolver A solely depends on cache I to looks up the servers, while resolver C simultaneously uses both cache I and non-cache lookups. On the other hand, resolver D does not use cache I at all and only directly look up the servers.

The variety of access methods from the resolvers to the servers of using or not-using the caches indicates that a resolver may refer to older contents if it is not explicitly told to *directly* look up the authoritative server *without using a cache*. While setting the TTL (Time-To-Live) of and RR to zero prevents the RR from being cached, clearing TTL of RRs results in the increase of processing load of the servers and reducing the efficiency gained by placing a cache because of the increase of end-to-end tendency between the resolvers and servers. Unnecessary reduction of TTL values should be avoided as much as possible to prevent increase of the server processing load and the number of payloads exchanged between the servers and resolvers, although a study [46] shows that the widespread use of dynamic, low-TTL A RR bindings should not greatly increase DNS-related wide-area network traffic.

<table>
<tr><td>Header
(flags and RCODES, etc.)</td><td>including the flag for
payload truncation</td></tr>
<tr><td>QNAME
(queried domain name)</td><td rowspan="2">query
information
(also included in
answer payload)</td></tr>
<tr><td>QTYPE and QCLASS
(queried RR data type and class)</td></tr>
<tr><td>answer section RRs
(directly answerable RRs)</td><td rowspan="2">answer
information
(included in
all answers)</td></tr>
<tr><td>authority section RRs
(non–direct reference RRs)</td></tr>
<tr><td>additional section RRs
(supplementaty information)</td><td>non–mandatory
answer section</td></tr>
</table>

Figure 2.5 DNS payload format

Table 2.1 An example of domain name compression

When `osaka-u.jp` is represented as follows (12 bytes):

offset	20	21	22	23	24	25	26	27	28	29	30	31
value	7	‘O’	‘S’	‘A’	‘K’	‘A’	‘-’	‘U’	2	‘J’	‘P’	0 (end)

offset	32	33	34	35	36	37
value	3	‘W’	‘W’	‘W’	PTR 20	

→ representing `www.osaka-u.jp` compressed from 16 to 6 bytes

quoted literals (‘A’): character A of a label (dot-separated word) in a domain name
(note: DNS literals are *not case-sensitive*)
number only (2): length of a label, zero (0) means the end of the name
PTR and *offset*: 2-byte offset pointing the domain name starts from the offset value

2.2.4 DNS Transport Specification

In this section, the author describes the behavior and requirements of DNS transport protocol, focused on the application payload format between the servers and resolvers, and the usage of lower-level transports, TCP and UDP.

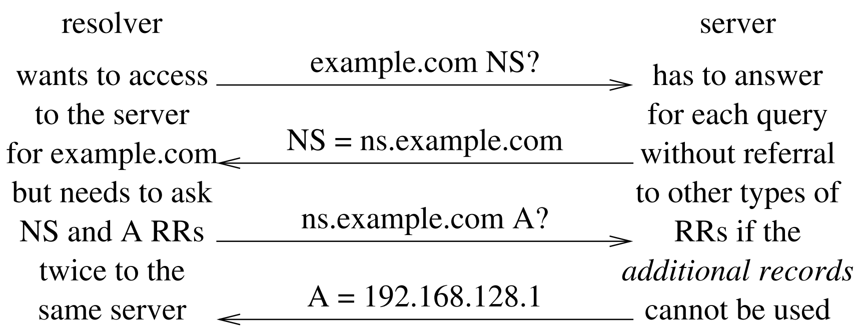

Figure 2.6 DNS name resolution without additional records

2.2.4.1 DNS Payload Format

Figure 2.5 shows the outline structure of DNS payloads. RFC1035 Section 4.1 defines the format of payload (called *message* in the RFC), and each payload makes a single query, request or answer. Each payload is carried in either a single UDP datagram or a single TCP connection.

The following is an explanation of the payload format:

- The header section is always present and includes fields to specify the number of other sections, and flags containing the control information, such as the `TC` (Truncation) bit, explained in Section 2.2.5.
- The question section includes the query information to the name server, with the fields of a query type (QTYPE), a query class (QCLASS), and a query domain name (QNAME). For Internet protocols, the query class has the fixed value of `IN`. Some of the common query types are listed in Table 1.1. The QNAME is a DNS-specific format of domain name string, which must use the compression of a domain name using 2-byte pointers for previously-appeared upper-level part of another domain name, as shown in Table 2.1 (RFC1123 [4] Section 6.1.2.4).
- The answer section contains the RRs answering the question. If the answer does not exist, the section contains no RR.
- The authority section contains the RRs that point towards one or more authoritative name servers, especially when the answering DNS server does not have an authoritative answer for the question.
- The additional section contains the RRs which relate to the query, but are not strictly answers for the question. A proper use of additional section RRs will reduce the number of queries.

  Figure 2.6 shows an example of a query and answer session without using additional

resolver		server
wants to access to the server for example.com and obtains the answer at once	example.com NS? → ← NS = ns.example.com A = 192.168.128.1 (for ns.example.com)	can answer to the query with some *additional records* to help faster name resolution

Figure 2.7 DNS name resolution *with* additional records

RRs for a query of NS RR, which needs two exchanges between the server and resolver. On the other hand, Figure 2.7 uses an additional RR to send the A RR to tell the address associated with the NS RR of the required answer, so only one exchange is required instead of two.

RFC2181 [5] Section 10.3 specifies that additional section does not include any aliases represented by CNAME RRs or A RRs associated with the CNAMEs. This means use of CNAMEs should be avoided whenever possible.

Note that some DNS implementations misinterpret *response* payloads as *query* payloads [47]. This may result in a message bouncing between servers and resolvers and cause a query-response storm, which is a form of DoS attack.

2.2.4.2 Recursive and Non-recursive Name Resolution

To resolve a domain name for an RR, a program must recursively perform the name resolution or server lookup to reach the authoritative server for a queried domain name, from the Root Zone to the lower-level domains.

Figure 2.8 shows an example of the recursive query for the A RR(s) of `www.osaka-u.jp` when no previous information has been given to the resolver. The resolver first asks to the Root Servers, the authoritative servers of the Root Zone, to find out the names (and IP addresses) of the Zone `.jp`, which is one level lower than the Root Domain. The resolver then tries to find out the names (and IP addresses) of the authoritative servers of one-level-lower domain, `osaka-u.jp`. Since an authoritative server of `osaka-u.jp` presumably knows the information of `www.osaka-u.jp`, which is guessed from the domain name syntax, the resolver finally sends a query for the A RR of `www.osaka-u.jp` to the authoritative server.

In Figure 2.8, the author assumes that the resolver is solely responsible for doing the recursion by itself. In the actual DNS programs, however, some server-like programs such as

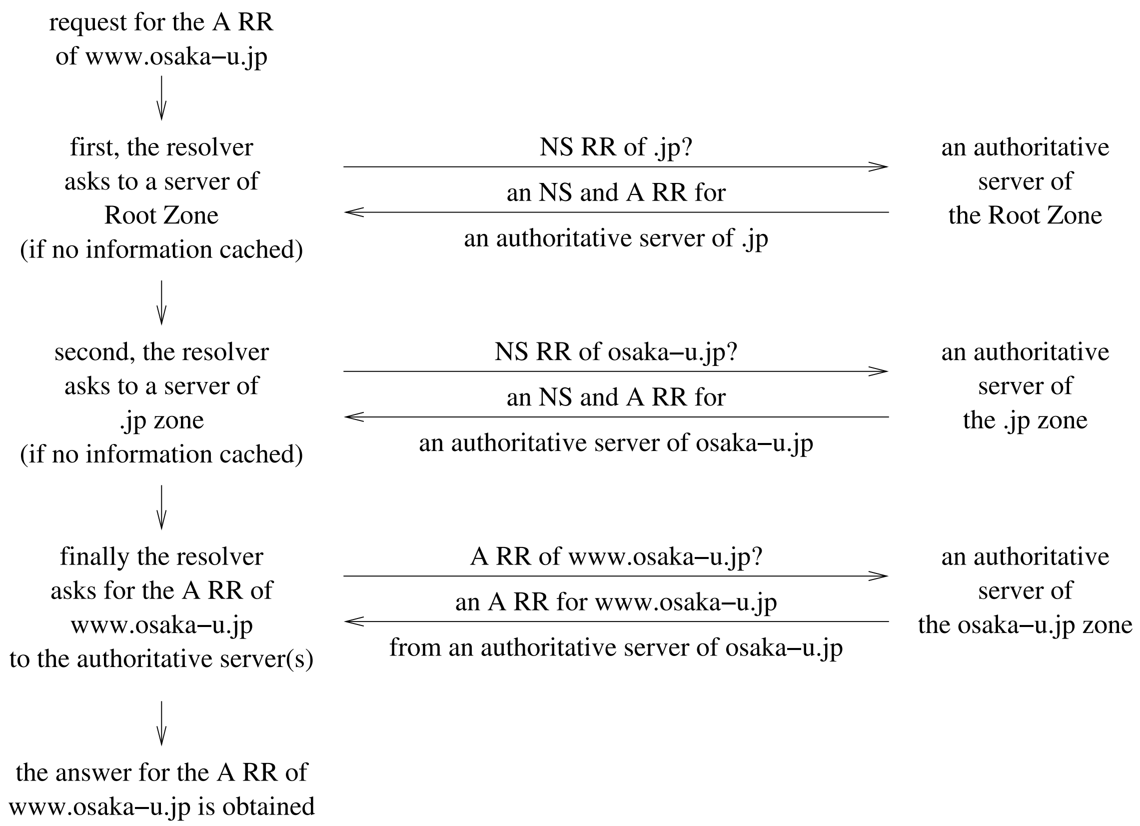

Figure 2.8 DNS name resolution using recursive queries

caches should be able to perform the necessary recursion process, to simplify the function of resolver by removing the recursion function. In fact, most client computers use this simplified version of resolver called *stub resolver*, since many existing servers such as those of BIND have traditionally been acting as caches for other servers. So the functionality of recursive query is negotiable between the resolvers and servers, and is only allowed when the server accepts the use. Some programs, such as `tinydns` of `djbdns` which only acts as an authoritative server for a limited number of pre-configured zones, only accepts non-recursive queries to prevent DoS attacks of queries to non-authoritative zone information.

2.2.4.3 DNS Transport Usage

DNS has two major forms of data exchange between the servers and resolvers, described in the Section 4.2 of RFC1035 [3], as follows:

Zone Transfer: this occurs between two servers for replication of Zone, a set of RRs which belongs to a domain name hierarchy, to obtain redundancy against a possible server failure and to prevent disruption of availability of the RRs in a zone.

The zone transfer is performed solely over TCP, since the size of a zone information set is much larger than a size of UDP payload, varying from several kilobytes to a few megabytes. The zone transfer can be used among the *authoritative* servers, so the transfer must be reliably performed.

RR Queries: this occurs between the servers and resolvers, to request and retrieve an RR for a domain name. Most of the real-world traffic of the queries is over UDP, though TCP is also allowed and supported by the majority of servers.

Some exchanges of control messages between servers, such as those of DNS NOFITY, also use the form of RR query from the sender to the receiver of the message.

Technical details of the DNS transport functions are also defined in RFC1123 [4], which defines host requirements connected to the Internet, and in RFC2181 [5], which clarifies the DNS specifications.

In this dissertation, the author mostly discusses the RR query issues in the later sections, since optimizing the resolver-server transaction for the larger payloads is the primary goal of the research. The author will not discuss the details of the zone transfer, since it is functionally the same as a file transfer over a single TCP link and has no limit of maximum size of transfer for each connection, and it will not degrade the performance of the resolver-server transaction.

2.2.5 UDP/TCP Choice: The 512-byte UDP Limitation

Section 4.2.1 of RFC1035 explicitly restricts the size of UDP queries and answers to 512 bytes. Section 6.1.3.2 of RFC1123 shows that a DNS server *must* service UDP queries and it *should* service TCP queries, and allows private agreement of servers and resolvers to solely use TCP for the queries.

2.2.5.1 Payload Truncation and UDP/TCP Limitations

Section 4.1.1 of RFC1035 specified the DNS header format. In the format, the `TC` bit is set when a server sends a truncated reply, due to the length greater than that permitted on the transport. The suggested behavior of the resolver which receives a UDP answer with the `TC` bit set is to reissue the request to the server using TCP [48] all over again as shown in Figure 2.9, although not all the implementations strictly comply with this sequence. This means that the query reply longer than 512 bytes is always sent back by TCP, after waiting a UDP exchange solely for the notification purpose.

For each UDP queries and answers, the length must be fit into a UDP datagram, which is 9216 bytes on FreeBSD 4.10-RELEASE. The practical length, however, is restricted by the size of maximum link-layer packet, since exceeding the size of the link-layer packet results in

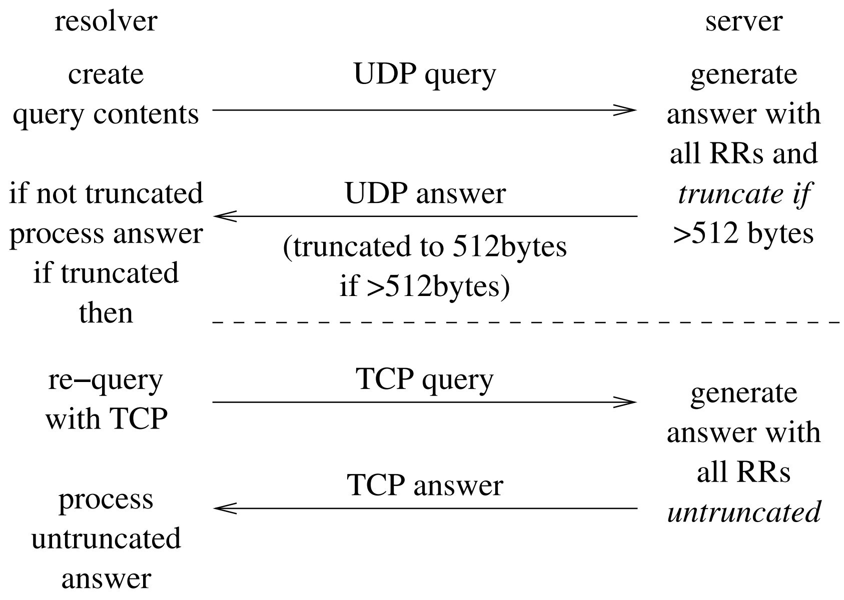

Figure 2.9 DNS resolver-server protocol fallback sequence from UDP to TCP

the fragmentation of the UDP datagram into multiple link-layer packets, and could degrade the reliability of application-level exchange.

For each TCP queries and answers, the length of the payload represented by a 16-bit unsigned integer is attached at the beginning of the actual query or answer being exchanged. On IPv4, the maximum length of TCP payload allows the maximum length of 65533 bytes for a DNS query and answer.

2.2.5.2 UDP/TCP Retransmission Strategies

DNS programs has its own retransmission and timeout algorithms for the UDP transport, since UDP does not retransmit by itself. Using TCP instead of UDP eliminates the need of designing the retransmission strategy, but at the cost of increase of minimum numbers of packets for each exchange of DNS payloads.

For example, `djbdns` uses the timeout algorithm [49] of waiting 3, 11, and 45 seconds respectively for each UDP recursive queries, and terminates the operation if nothing received after retransmitting three times. This retransmission strategy works well when the packet loss rate of the network is small. When the packet loss rate is very high, however, it may cause delay of the completion of query processes, either succeeded or failed, since only four or less packets are sent for each query.

On the other hand, BIND Version 8.3.7-REL resolver library included as a part of FreeBSD 4.10-RELEASE, retries usually only twice and 5 times at maximum, with a fixed length of interval time, usually 5 seconds and 30 seconds at maximum, configurable by the caller of the library functions. While this strategy works well when the latency of the network is small, the fixed-length interval may cause network congestion when the number of resolvers for a server is very large.

2.2.6 The Root Server's Example of 512-byte UDP Limitation

One of the most important examples which the UDP payload length limit affects the system design of DNS is the case for the Root Servers. To avoid processing burdens being caused by TCP queries, answers to the DNS queries for the Root Servers must be fit into 512 bytes. The UDP size limitation means a restriction which only 13 IPv4 servers can be specified in the SOA answer for the Root Domain, of 1 SOA, 13 NS and 13 A RRs, 493 bytes in total.

The limitation of the Root Zone RRs which can be included in a payload significantly hampers the necessary change for the growth of DNS, such as increasing the processing performance of the Root Server network by adding more addresses for load balancing and distribution, and introducing IPv6 addresses to the Root Domain for the migration. If the number of Root Servers were increased or some of the servers also announced the IPv6 addresses by the AAAA RR, the answer could easily exceed the 512-byte size limit [48], so the query reply for the Root Servers would not be able to be carried over UDP. The details of this problem is also discussed in Section 3.5.7.

2.3 Support for Migration from IPv4 to IPv6

DNS must support IPv6, as Internet networks and hosts are currently on the way of migration from IPv4 to IPv6. In this section, the author describes some of the key issues for DNS to support migration from IPv4 to IPv6.

The major change for DNS RRs to support IPv6 is introduction of AAAA RR and `ip6.arpa` domain for reverse lookups, described later in Section 3.2.1. Introduction of IPv6, however, creates many operational considerations and issues to solve, for many reasons including, but not limited to, the following:

- IPv6 network is, on the contrary from the popular belief, an independently-built network from the IPv4, although many of the characteristics are common;
- Core Internet services including DNS *must* support the IPv6 objects and guarantee the same level of accessibility and availability of services as well as those of IPv4; and
- IPv4 and IPv6 coexists during the transition period, although many hosts do not support IPv6 services for DNS yet, so they will try to access IPv6 information through IPv4, and

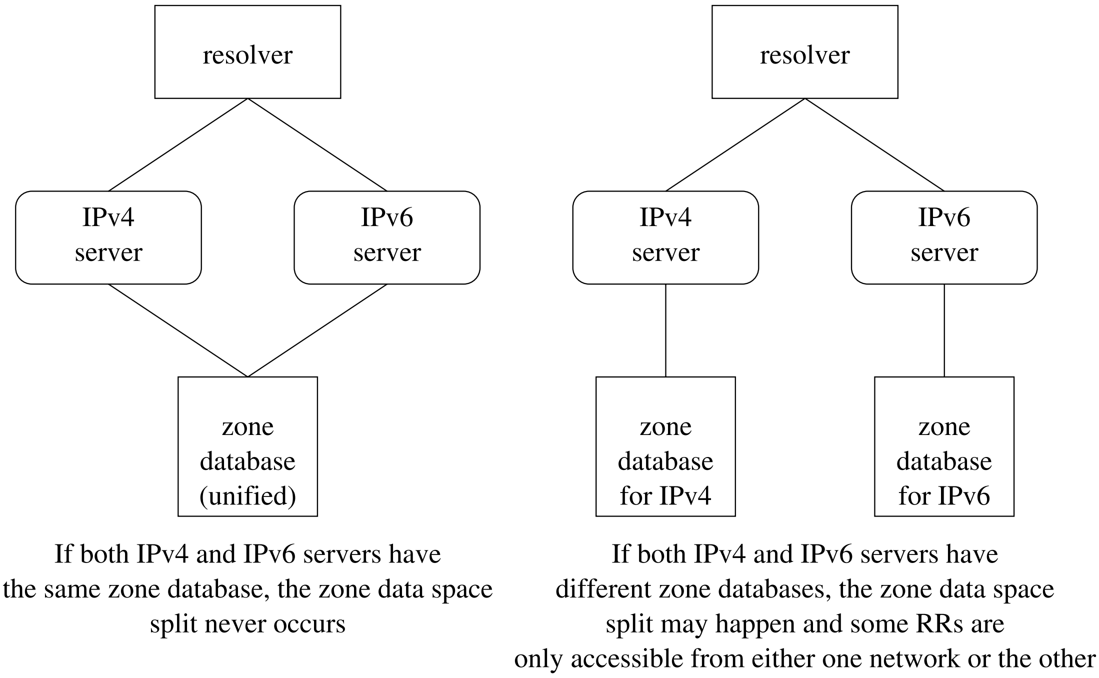

Figure 2.10 Resolvers may see split zone data spaces through IPv4 and IPv6 networks

vice versa as well in the future when IPv6-only hosts take the majority of the Internet.

2.3.1 IPv4/IPv6 Split Zone Data Spaces

An example of one of the problems is *split zone data spaces* between zone data spaces which can be seen from IPv4 and IPv6, also called as *IPv4/IPv6 name space fragmentation* [50, 51]. Since the accessible zone data through IPv4 and IPv6 are not necessarily the same, some IPv6-only host may not find out a DNS RR which can be found through the access via IPv4.

Figure 2.10 shows a case of database unification problem for this issue. Completely unifying the DNS database throughout IPv4 and IPv6 DNS servers is virtually impossible, since some servers may only contain IPv4-specific RRs, while other servers may only contain IPv6-specific RRs, so the referral chains between the servers may easily collapse when the communication network does not exist between two specific servers, because of lack of support of IPv4 or IPv6. Durand and Ihren [51] suggests the two administrative policies should be implemented to avoid the split zone data spaces:

- every recursive name server should be either IPv4-only or dual stack (supporting IPv4 and IPv6); and
- every DNS zone should be served by at least one IPv4-reachable authoritative name

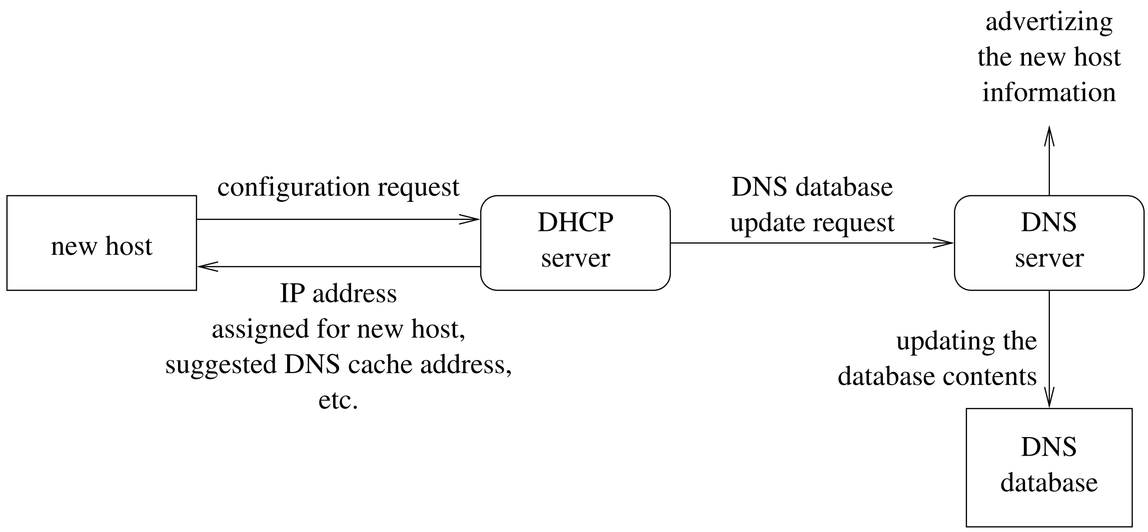

Figure 2.11 Updating DNS database from DHCP server

server.

With the policies mentioned above, DNS administrators should maintain the current practice that *all* DNS name spaces are accessible through the IPv4 network. IPv6-only hosts or stub resolvers can access to the IPv4 DNS servers through a dual-stack DNS cache, so this will not cause a major operational problem.

2.3.2 Autoconfiguration and Updating DNS Database

IPv6 also enforces autoconfiguration of host address, in both stateful and stateless mechanisms. The stateful configuration means DHCP (Dynamic Host Configuration Protocol) for IPv6 (DHCPv6) [52], as well as that for IPv4 [53], which each host asks the configuration information to a DHCP server at the boot time. The stateless configuration [54] means that the host automatically defines the IPv6 address at the boot time using only the internal hardware-related information and no external configuration for determining the IPv6 address.

Figure 2.11 shows an example of interaction between the host to be configured, DHCP server, and DNS server. In either case of stateful or stateless configuration, interaction with DNS should be done for providing the configuration service, including the following actions:

- notifying the suggested DNS non-stub resolver or cache address to the configured host (usually from a DHCP server) for external DNS access; and
- updating the DNS zone database after authenticating and fixing the stateful or stateless address of a host.

The details on updating DNS dynamically are discussed in Section 2.5.

2.4 Authentication of RRs and Payloads by DNSSEC

IP and upper-layer transports such as UDP and TCP do not provide protection against forgery or alteration of the contents, which is a problem for DNS since forged RRs can be abused for attacks such as redirection to malicious hosts or wiretapping of electronic mail messages. Protecting DNS payloads and RRs from alteration or forgery attempts is crucial, while hiding DNS payloads is not necessarily a requirement due to the public nature of DNS RRs, configured for public access throughout the wide-area Internet.

Authenticating the DNS payloads and the RRs has become a key development goal for many years. DNSSEC [8] is the primary extension for cryptographic per-RR authentication within the DNS transport protocol, while cryptographic authentication on other layers such as IPsec is also under development and deployment. In this section, the author describes DNSSEC and its ongoing development status, and the limitation.

2.4.1 Past DNSSEC and The Limitation

As of December 2004, DNSSEC is still considered as a primary means to secure the DNS, though the first protocol design has become historic and will no longer be widely deployed.

When it was first designed in 1999, basic elements of DNSSEC were as follows:

- KEY RR, to distribute a (public) key associated with a DNS or domain name;
- SIG RR, to provide signature for an RRset (set of RRs);
- NXT RR, to show non-existence of a name in a zone; and
- the requirement that a child zone needs to have its KEY RRs signed by its parent.

Using those basic elements, a zone-level authentication is performed by the public-key cryptographic system, by making a chain of trust as the same path of zone delegation.

Two other signature schemes are proposed: TSIG in RFC2845 [41], and SIG(0) in RFC2931 [55]. While SIG RR authenticates an RR of a DNS Zone with a public-key cryptographic system, TSIG and SIG(0) authenticates each transaction. TSIG uses a shared-key cryptographic system, while SIG(0) uses a public-key cryptographic system. While TSIG and SIG(0) are primarily designed for protecting secure DNS update requests, Baba et al. [56] propose an implementation using SIG(0) to authenticate the host or user of the resolver for the access control of the DNS database.

DNSSEC, however, does not provide the protection against DoS attacks, as described in RFC2535 Section 2.1. For preventing DoS attacks, it is required to use other means than DNSSEC, such as those for the TCP, UDP or other TCP/IP transport layer protocols.

The author considers that the feasibility of wide-range DNSSEC deployment is low in the above signatory schemes, while he is convinced that authentication of DNS RRs is crucial and will become mandatory for future DNS, because of the following reasons:

- TSIG is a shared-key system, and for the implementation, the key-distribution security of the secret key has to be maintained. This will not work for multilateral inter-organizational system such as the global Internet. Even using a public-key system such as SIG(0) or SIG, millions of public keys have to be maintained for each second-level domain name.
- SIG and SIG(0) uses a public-key system, which is computationally resource-intensive, and may impact the overall performance of DNS. For distributing the public keys whose digits are long enough for giving enough protection, the length of RRs will increase and may exceed the limit of 512 bytes for DNS UDP exchange. This may also hamper the DNS performance as a whole.
- The authentication model of DNSSEC assumes that the communication is performed directly between the resolvers and the servers. In a practical DNS configuration, however, the resolvers use caches and indirectly exchange information between the servers. In this cached model, TSIG and SIG(0) cannot provide the end-to-end authentication between the resolvers and the servers. A similar problem may occur when handling a replicated Zone data by DNS zone transfer.

2.4.2 DNSSEC based on Delegation Signer (DS)

DNSSEC has been under ongoing change from the past SIG-based scheme to a new scheme using Delegation Signer (DS) RR [57]. The DS changes the model as follows:

- The zone administrator can sign the zone itself by using the Zone Signing Key (ZSK);
- Key Signing Key (KSK) is introduced to sign the ZSK;
- DS RR, which contains the digest of a public key that is allowed or used to sign the child zone's KEY RRset, is introduced; and
- The parent zone authorizes the KSK of a child zone.

Figure 2.12 shows a DNS name resolution process of `www.osaka-u.jp`, similar to that in Figure 2.8, but each record returned from the authoritative servers is with corresponding DS RR and the signature for DS RR. The chain of DS RRs enforces authentication of the chain of

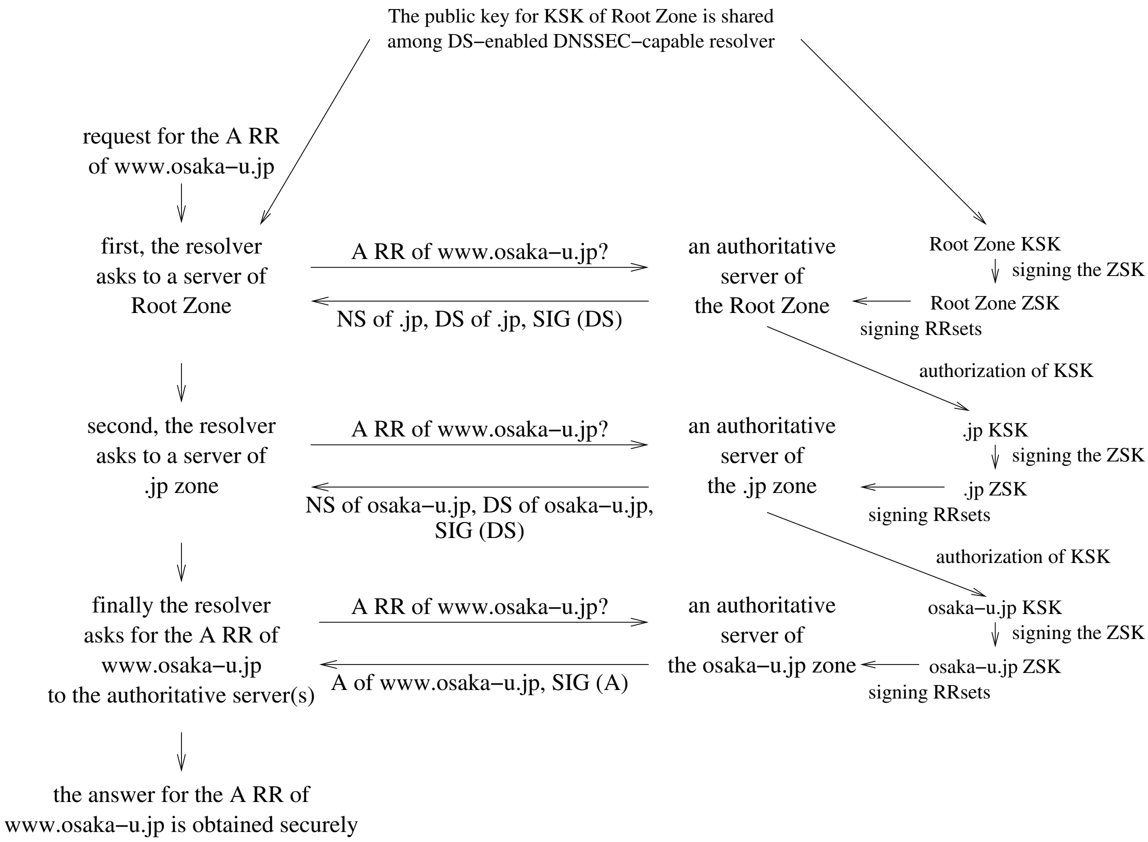

Figure 2.12 DNS name resolution with DS-based DNSSEC (based on a JPRS document [58])

NS RRs. The final A RR is with its SIG RR signed by the ZSK of `osaka-u.jp`, which has the final authority for the name `www.osaka-u.jp`.

DS-based DNSSEC is *not* compatible with the past DNSSEC using SIG and KEY records, so the servers, caches and resolvers must be updated for the change. BIND Version 9.3.0 and NSD 2.1.5 claim support for the DS-based DNSSEC. The key distribution and update issues, which are common problems to deploy public-key-based systems, are still needed to be externally solved.

2.5 Dynamic Update of DNS Contents

Dynamic update of DNS contents is essential to support DNS mapping to mobile hosts which moves around different networks, and to reduce the administration overhead of maintaining one-by-one IP address assignment to large number of client hosts, whose IP address and security policy management is sufficient by treating them as a group and allowing IP address change of each host.

DNS UPDATE [9] defines a DNS content update extension to add or delete RRs from

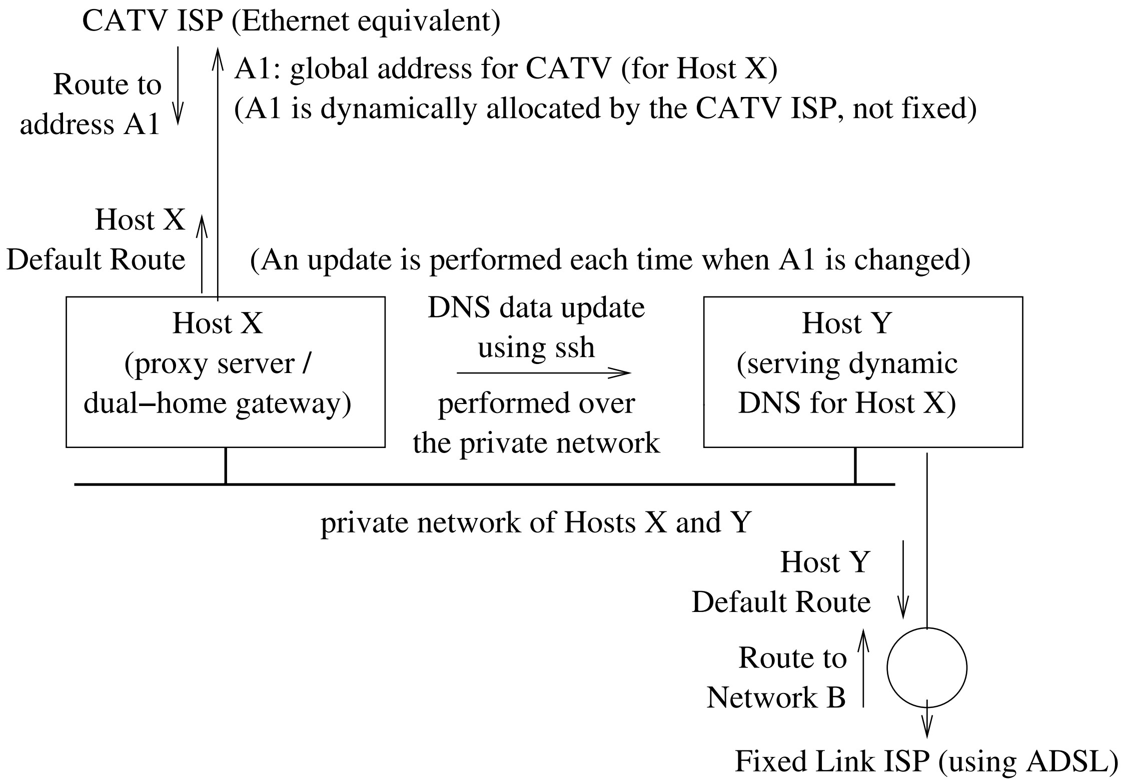

Figure 2.13 An example of dynamic DNS update using ssh

a specified zone, with prerequisites specified, and can specify a dependency upon either the previous existence or non-existence of an RRset, or the existence of a single RR.

The operation of updating the zone database is performed by using the UPDATE opcode which indicates the updating request, with the similar format of DNS payloads for the RR queries and answers. The payload length issues of DNS RR queries and answers are equally applicable to the UPDATE opcode.

The updating request of DNS UPDATE should be sent to the *Primary Master* server, which is the master of all the authoritative servers for the zone, whose name is in the zone's MNAME field of SOA RR. Only one primary master server per zone is allowed.

To secure the update operation against forgery and alteration attacks, Secure DNS Dynamic Update [28] is proposed as an enhancement to DNS UPDATE. It incorporates the signature for entire payload or message using TSIG or SIG(0) secure transaction schemes, and requires the use of signed UPDATE opcodes for all updating requests. While DNSSEC authenticates each RR, Secure DNS Dynamic Update authenticates each updating requests and answers.

Updating the DNS contents, however, are not only performed through (Secure) DNS UP-

DATE. Dynamic DNS [59] uses HTTP or Secure HTTP requests to update A RR and MX RR of a host name, and has been a popular service among non-fixed-IP address hosts, which consists of the majority of dialup- or broadband-network-connected hosts. By providing a fixed name for the same user while the IP address may change, the user can open her/his own servers to the public without obtaining the fixed IP address by him/herself. Using (Secure) HTTP, the request can be sent from networks protected inside firewalls running an HTTP proxy.

Figure 2.13 is another example of updating DNS database using dynamically-assigned IP address, running at the author's home since November 2001 [60, 61]. Host X in Figure 2.13, which has the dynamically-assigned global IP address, explicitly performs the data update by rewriting the database of a DNS server running in Host Y, using Secure Shell (ssh) protocol based on OpenSSH [62] with the public-key authentication scheme. Host Y advertizes the A RR for Host X so that Host X is accessible through the assigned address from the CATV ISP. While this is not a solution for handling many dynamic-IP hosts, it is a practical solution for a small system to add a globally-accessible address at a minimum cost.

2.6 Concluding Remarks

In this chapter, the author described the architectural issues of DNS and defined the transport protocol being analyzed and evaluated in this dissertation, and described the emerging new issues such as migration support from IPv4 to IPv6, RR and payload authentication using DNSSEC, and dynamic update of DNS database.

Since the role which DNS plays becomes more and more versatile and complex, the payloads it handles also become more complex and of larger size. While the system design of Root Servers has already been affected as shown in Section 2.2.6, the number of RRs exchanged in a single payload continues to increase, due to many reasons including addition of signature RRs by DNSSEC, transactional authentication of each payload by TSIG and SIG(0), and the simple increase of the complexity of systems connected to Internet.

The payload length will also increase as IPv6 becomes more popular. The cost of having IP addresses allocated becomes much lower than that in the current limited IPv4 address space, so more AAAA RRs will have to be exchanged in a single payload. The details are addressed in Chapter 3.

The reliability of DNS exchange will become a more critical issue, as dynamically modifying the DNS database contents becomes popular. Exchanges for DNS database update should be protected with digital signature and a reliable lower-layer transport protocol such as TCP to ensure the atomic transaction. On the other hand, the overhead of TCP is significant comparing to that of UDP on operating a large-scale system using dynamic DNS update, so a faster reliable protocol is desired.

In the following chapters, the author addresses the two specific problems, DNS payload length increase and a faster reliable transport protocol implementation for DNS, through the analysis of simulation of migration to IPv6, and the evaluation of implementing T/TCP to DNS programs.

Chapter 3

DNS Payload Length Increase during Transition to IPv6

3.1 Introduction

The DNS transport protocol, which handles the query-and-answer exchange between DNS resolvers and servers, is designed upon an assumption that the length of data exchanges in each query does not exceed a few hundred bytes. Under this assumption, the 512-byte limit of UDP payload and the protocol fallback to TCP for a larger payload are imposed.

While the current assumption of DNS transport payload length works well on the current Internet infrastructure mostly based on IPv4, the recent protocol enhancement trends, such as the migration to IPv6, increase the length of payload and also increases the percentage of the payloads larger than 512 bytes. For example, AAAA RRs, representing a 16-byte IPv6 address for a domain name, will become a major portion of queried RRs instead of A RRs, representing a 4-byte IPv4 address, which currently takes a major portion of DNS database answers. This indicates that the payload length of DNS answers will increase as Internet migrates from IPv4 to IPv6.

The 512-byte payload limitation has already become a major operational issue of DNS. The maximum number of Root Servers, the authoritative servers of the Root Zone information, is only 13, due to this limitation. Increasing the number results in generating massive TCP traffics being fallen back from the failed UDP requests, and is not practical. The IPv6 migration of Root Servers will result in the same manners due to the larger requirement of data length to represent the addresses. This issue is also applicable to all large-scale DNS servers serving the same zones.

In this chapter, the author discusses the issues which arises during the migration to IPv6 due to the DNS UDP payload length limitation, and quantitatively analyze how the issues affect the DNS traffics by simulating the payload length of added or changed RRs during and after the

Table 3.1 Change in PTR RRs from IPv4 to IPv6

IPv4	`111.222.123.234` → `234.123.222.111.in-addr.arpa`
IPv6	`0123:4567:89ab:cdef:1213:2324:3435:4647` → `7.4.6.4.5.3.4.3.4.2.3.2.3.1.2.1.f.e.d.c.b.a.9.8.7.6.5.4.3.2.1.0.ip6.arpa`

migration from IPv4 to IPv6, by using the real-world DNS database traffic data [63, 64]. The author then proposes possible solutions such as EDNS0 [6], a DNS protocol enhancement, and comparatively evaluate the levels of improvement for each solution.

In the later sections, the author describes the details of increasing trend of DNS payload length on Section 3.2, and the methodology to perform real-world DNS traffic analysis on Section 3.3. On Section 3.4, the author explains the details of the simulation conditions and algorithms used in the real-world DNS traffic analysis, and on Section 3.5 the author evaluates possible solutions for handling the larger payload length. The author concludes this chapter on Section 3.6 with a discussion of a roadmap for implementing DNS protocol enhancements to handle the larger-length payloads.

3.2 The Increasing Trend of DNS Payload Length

In this section, the author describes the trend and causes of increasing length of DNS transport protocol payloads and the transferred RRs, and how the trend affects the DNS traffic behavior.

3.2.1 Change of Type and Increase of Length of RRs Due to Migration to IPv6

The following issues also shown in Figure 3.1 should be considered for change of type and increase of length of RRs during the migration from IPv4 to IPv6:

- On DNS, an AAAA RR (RFC3596 [65] Section 2) is used for a reference of an IPv6 address from a domain name. In this chapter, the author solely focuses on AAAA RRs to represent IPv6 addresses. The other proposed address resolution method using A6 and DNAME RRs [66] is now considered *experimental* and the AAAA RRs are considered *preferable* for the production deployment of IPv6 (RFC3363 [67] Section 2) after an extensive discussion in IETF `dnsext` and `ngtrans` working groups [68].

  As the migration from A RR to AAAA RR proceeds, the length of each RR representing

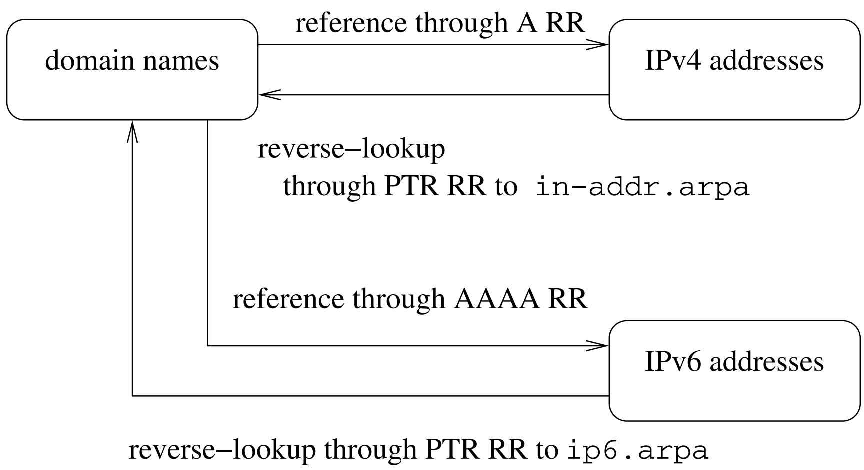

Figure 3.1 Change of RR reference methods by the migration from IPv4 to IPv6

IP address increases. While A RR for IPv4 (RFC1035 Section 3.4.1) defines 32-bit (4-byte) address value to the RDATA field for a domain name, AAAA RR for IPv6 defines 128-bit (16-byte) address value instead.

- To perform *reverse lookups*, which is references from IP addresses to the corresponding domain names, the domain `in-addr.arpa` is used for IPv4 addresses (RFC1034 Section 5.2.1). On IPv6, the domain `ip6.arpa` (RFC3596 Section 2.5) is used instead.
  The length of PTR RRs for reverse lookups gets longer as the migration from IPv4 to IPv6 proceeds. Table 3.1 shows that while the reverse-lookup domain name length for IPv4 is 28 bytes in maximum, the length increases to 72 bytes maximum for IPv6.

As the length of RRs increase, the author predicts that the following changes will occur for the queries and answers over DNS transport protocol:

- The length of RRs for IP-address lookups will change during the migration from IPv4 to IPv6, proceeding through the phases of IPv4-and-IPv6 coexistence and the completion of the migration to IPv6 and the phasing out of IPv4, as follows:
  - When an A RR for IPv4 address is replaced by an AAAA RR for IPv6 address, the length increases by 12 bytes as the address length changes from 32 bits to 128 bits.
  - To add a RR of IPv6 address for an IPv4 host, the length of RRs for the IP v4-and-v6 addresses of the host increases by at least 28 bytes. The increased portions consist of the 2-byte domain index information compressed as described in RFC1035 Section 4.1.4, 10-byte header for the AAAA RR, and 16-byte RDATA of the AAAA RR which contains the IPv6 address.

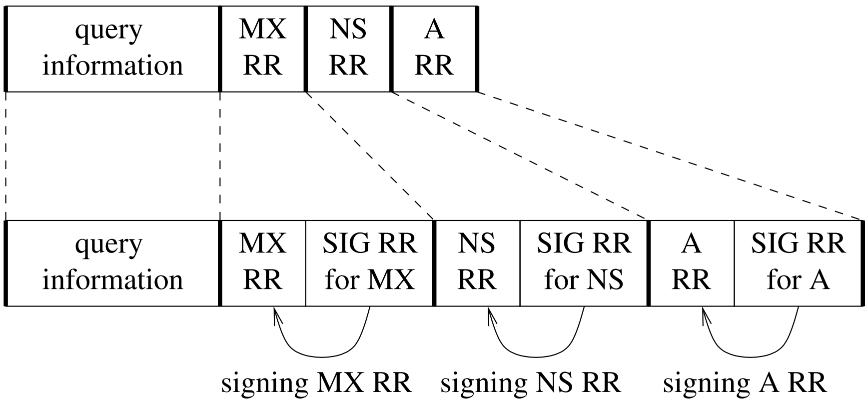

Figure 3.2 How each RR is signed by the corresponding signature RR on DNSSEC

- The change of reverse-lookup namespace from `in-addr.arpa` to `ip6.arpa` will result in the length increase of the payload, which contains the answer for the reverse-lookup by 48 bytes maximum for each reverse-lookup query. This will not affect much on the cumulative payload length, however, since the length of the domain name used for the reverse lookup is compressed to 2-byte index information as described in RFC1035, and the length of PTR RRs of the answers does not change.

3.2.2 Other factors to Increase DNS Payload Length

The following two factors, other than the migration to IPv6, contribute for increase of DNS Payload Length. These factors are not direct results of the migration to IPv6, but are the results of DNS functional enhancements and ongoing change of Internet usage, and should be considered for finding out the future direction of DNS transport protocol.

- The introduction of DNSSEC [8] requires each RR to have an additional public-key signature RR (SIG RR, shown in Figure 3.2). In RFC3226 [48] Section 2.1, the length for the signature RR is predicted between 80 to 800 bytes, and most of the RRs are equal or less than 200 bytes.
- It becomes more common that DNS servers return multiple numbers of RRs for a single query is increasing. For example, in the Web virtual domain service, the same IP address is shared for multiple domain names. In this case, a reverse-lookup request for a virtual-domain IP address causes all corresponding domain names to be returned as multiple PTR RRs. Another common example is to randomly return multiple IP address values for a DNS query of a Web server domain name, for balancing the processing load throughout

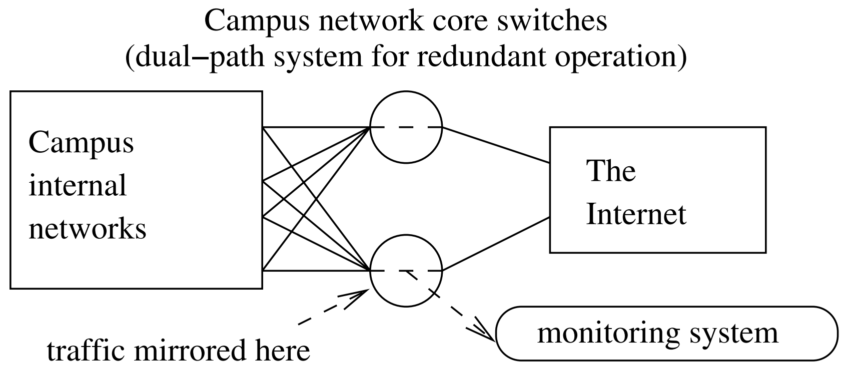

Figure 3.3 System diagram for collecting DNS traffic

multiple hosts. These practices cause increase of the number of RRs to be returned to a single DNS query.

3.2.3 How the DNS Payload Length Limitation Affects the Root Zone

In the current DNS operation, answers from the Root Servers, the set of DNS servers which provides the zone information of the Root Zone, the top level of the DNS hierarchy, are most affected by the DNS UDP payload length limitation of 512 bytes.

Currently the domain names of Root Servers have 13 names under `root-servers.net` domain such as `a.root-servers.net`, whose third-level symbol is from `a` to `m`. Under the current limitation of 512 bytes, no more Root Servers can be added, due to the operational recommendation to restrict TCP queries to the Root Servers as little as possible for preventing the processing load increase. By this limitation, neither new server nor new IPv6 address can be added or assigned to the Root Servers. The author discusses the detail of this issue in Section 3.5.7.

3.3 Method of Real-World DNS Traffic Analysis

In this section, the author describes the method to collect the real-world traffic data of DNS and the analysis of the collected data in details, for the further simulation of DNS payload length change during the migration from IPv4 to IPv6.

Table 3.2 Numbers of DNS answers collected for the analysis

starting and ending times of each measurement (JST: Japan Standard Time)	numbers of answers		TCP/UDP (%)
	TCP	UDP	
28-NOV-2003 0836~2035JST	7387	6249736	0.118
16-DEC-2003 0047~1246JST	4581	2997881	0.153
Total	11968	9247617	0.129

Table 3.3 Numbers and percentage of characteristics in collected DNS answers with TCP

Characteristics	28-NOV-2003		16-DEC-2003	
Single TKEY RR	6838	(92.6%)	4018	(87.7%)
Non-TKEY RRs of length >512	51	(0.7%)	61	(1.3%)
Non-TKEY RRs of $1 \leq$ length ≤ 512	341	(4.6%)	334	(7.3%)
Others	157	(2.1%)	168	(3.7%)

(length values are in bytes)

3.3.1 Collecting the Raw DNS Traffic Data

Figure 3.3 shows the diagram of the system used to collect the raw DNS traffic data. In this diagram, a monitoring system tapped into one of the two core switches which handled the traffic between inside and outside of a large-scale campus network, of Osaka University. Each core switch randomly forwarded the packets, and provided redundancy in case of a failure of either one of the switches. The switch connected to the monitoring system duplicated the packets using the mirroring function, and fed the mirrored traffic to the monitoring system. The monitoring systems was connected to the switch through a 1000BASE-SX optical Ethernet link, running FreeBSD [29] as the operating system and `snort` [69] as the packet-collecting software. UDP packets of non-zero fragment offsets were ignored during the analysis.

The traffic monitoring was performed twice in the JST (Japan Standard Time) morning of November 28th, 2003, and the late night of December 16th, 2003, each continued for 12 hours. The analysis of each DNS packet was performed by `tcpdump` [70] with modification of support detailed analysis of DNS-specific attributes.

3.3.2 Choosing the Data to Analyze

In this chapter, the author decides to analyze only the UDP packets over IPv4, whose source or destination port numbers are 53, assigned to the DNS protocol. The analysis is only performed on DNS answers, not on the queries. The following four list items are the reasons:

- In the real-world DNS operation, most of the DNS traffics are on IPv4, and the percentage of IPv6-only DNS traffic is very small. For example, Root Servers have no IPv6 address assigned. Two of the four authoritative servers of `ip6.arpa`, the IPv6 reverse-lookup domain, can only be queried through IPv4. A guideline of IPv6 DNS operation [51] suggests that all DNS servers should be capable to serve on IPv4 networks to prevent name space fragmentation between IPv4 and IPv6 DNS name spaces.
- The frequency of RR queries and answers by TCP is 0.12%~0.16% of that by UDP as shown in Table 3.2. About 9 out of 10 RR answers by TCP are TKEY RR [71], used for secret-key exchange of DNSSEC TSIG authentication [41], as shown in Table 3.3. The number of all of the other RR answers were between 500 to 600 for each traffic collected twice, which is only 0.01%~0.02% of the number of total RR answers by UDP. The author concluded that the RR answers by TCP could be excluded from the simulation of the migration from IPv4 to IPv6.
- The length of DNS name must not exceed 255 bytes (RFC1035 Section 2.3.4). When a DNS name is converted as QNAME embedded in a DNS query, 2 bytes are added, so the maximum length is 257 bytes. The possible maximum payload length of a DNS query is 273 bytes, adding 12 bytes of the header, 2 bytes of QTYPE and 2 bytes of QCLASS fields (RFC1035 Section 4.1.2), much smaller than 512 bytes. This indicates that excluding DNS queries does not affect the evaluation criteria of whether exceeding 512 bytes or not for each DNS payload.

Table 3.4 shows the percentage for each type of RRs in collected DNS answers. Comparing the A and AAAA RRs shows that A RRs take more than 40% of the all RRs, while AAAA RR take only 0.8% to 2% of the all RRs. This indicates that the RRs for IP addresses take the major part of DNS answers, and that the migration to IPv6 has not much proceeded.

By collecting the payload length, the number and type of RRs contained for each DNS answer payload, a simulation which verifies the prediction in Section 3.2.1 can be performed.

Table 3.4 Percentage of RRs in collected UDP DNS answers

RR Type	28-NOV-2003	16-DEC-2003
A	40.17%	42.69%
AAAA	0.87%	1.95%
CNAME	0.94%	0.59%
MX	1.26%	1.73%
NS	35.64%	39.22%
OPT	16.32%	9.12%
PTR	0.84%	0.98%
SOA	3.96%	3.73%
Others	< 0.01%	< 0.01%
Number of total RRs	22642277	14460833

3.4 A Simulation of Transition Period to IPv6

In this section, the author predicts how the trend of DNS payload length will change during and after the migration from IPv4 to IPv6, by simulation of performing recalculation of the payload length to the real-world traffic data.

3.4.1 How the Payload Length Increase Is Simulated

The simulation of payload length increase is performed for the following two cases:

IPv4+IPv6 co-existing phase Assuming that each host represented by a domain name adds one AAAA RR for each existing A RR during the co-existing phase of IPv4 and IPv6, the payload length is increased by 28 bytes for each A RR in the collected DNS answer.

IPv6 migration-completed phase Assuming that each existing A RR is replaced by a newly-assigned AAAA RR for each host represented by a domain name after completion of the migration to IPv6, the payload length is increased by 12 bytes for each A RR in the collected DNS answer.

The above simulation process is performed over all A RRs included in the answer section and the additional section.

The simulation process does not consider the domain names which have already both A and AAAA RRs assigned. The number of as the statistics in Table 3.4 shows, however, that the number of A RRs in the collected DNS answers are more than 20 times of that of AAAA RRs, so the author considers performing simulation only on A RRs is sufficient to predict the overall change of DNS answers.

RFC2181 [5] Section 9 suggests that if the payload length exceeds the limitation and the `TC` bit of DNS header is set *solely because of* the RRs in the *additional section* which are not requires to be sent together with the (required) answer section, the RRset (set of RRs) that will not fit in the response should be omitted and the answer payload sent as is, with the `TC` bit cleared, to make the length below or equal to the limit value of 512 bytes.

In this chapter, the simulation process are performed in the two set of cases, whether ARs (Additional Records, RRs in the additional section of a payload) are contained or not contained in the answer payload.

On the actual DNS operation, complete removal of ARs may cause malfunction on DNS lookups [72]. The IP-address RRs contained as ARs are usually the addresses of NS RRs in the same DNS answer payload, and are essential to reduce the total number of queries for the address resolution. The author claims that the selective removal or choice of ARs should be considered as a operational issue under the 512-byte payload length limitation and that the ARs should be preserved as possible. In the later simulation and analysis, the author mainly focuses on the cases where ARs are fully contained in DNS answer payloads.

3.4.2 Analysis of The Simulation Results

Table 3.5 shows the statistics of collected DNS answers and the result of simulation by adding or replacing AAAA RR to A RR in the answers.

In either case removing or leaving ARs in the DNS answers, the mean value (μ) and the standard deviation (σ) are increased after the addition or replacement of AAAA RR. In the case when ARs are contained in the answer payloads, the percentage of payloads larger than 512 bytes is increased from less than 0.04% of the collected data to 1~3% after the simulation is performed. In the case when ARs are removed in the answer payloads, the percentage of payloads larger than 512 bytes is 0.001~0.002% of the collected data, about $1/10 \sim 1/20$ of that of the case when ARs are contained and becomes much smaller, but even in this case after the simulation is performed the percentage increases to 0.06~0.14%.

According to the simulation results, in either case removing or leaving ARs in the DNS answers, the percentage of payloads larger than 512 bytes after the simulation is increased to 20~100 times of that before the simulation.

Table 3.6 shows the classification of characteristics in RRs of collected UDP DNS answers, such as the answer of the queried server itself, referring to other servers, or other protocol

Table 3.5 Statistics for the simulation results

for 6249736 samples of 28-NOV-2003				
	μ	σ	max	>512
raw data w/o AR	81.80	52.73	1149	0.001
raw data with AR	108.26	79.68	1192	0.023
AAAA+A w/o AR	89.14	66.66	3025	0.136
AAAA+A with AR	149.01	142.28	3124	2.117
AAAA→A w/o AR	84.95	57.74	1953	0.075
AAAA→A with AR	125.72	105.98	2020	1.124
for 2997881 samples of 16-DEC-2003				
	μ	σ	max	>512
raw data w/o AR	102.28	53.57	944	0.002
raw data with AR	137.90	87.16	1112	0.035
AAAA+A w/o AR	110.58	66.02	1485	0.128
AAAA+A with AR	195.55	155.43	2285	2.772
AAAA→A w/o AR	105.84	57.82	973	0.064
AAAA→A with AR	162.60	115.83	1533	1.656

μ: mean value (bytes)

σ: unbiased standard deviation (bytes)

max: maximum payload length (bytes)

>512: % of payloads longer than 512 bytes

w/o AR: without Additional Records

errors. The percentage of the answers by the queried servers themselves is 20~28% of the total. The percentage of the reference to other servers is 28~39% of the total, and the author claims this indicates the difference of the payload length between the cases with or without ARs.

Table 3.7 shows the statistics of QNAME length in the collected DNS answers. The number of samples for December 16, 2003, is 14 smaller than that shown in Table 3.5. The 14 packets are illegal ones from the same host and has no valid QNAME field, so they are not counted as samples for the statistics.

The mean value of QNAME length is 22~23 bytes, 17~21% of the mean value of the whole payload length. This indicates that the major portion of DNS payloads are *not* QNAME but other answer records.

Table 3.6 Percentage of characteristics of RRs in collected UDP DNS answers

Characteristics	28-NOV-2003		16-DEC-2003	
Answer with authority (1)	1730603	(27.7%)	1070553	(19.5%)
Server errors (2)	1217000	(19.5%)	168369	(5.6%)
Referral to other servers (3)	1763301	(28.2%)	1164987	(38.9%)
Others	1538432	(24.6%)	593972	(19.8%)

(1) ancount>0 in the header, or RCODE in the answer shows NXDOMAIN or codes related to DNS UPDATE [9], which means that the server itself replies the existence or non-existence of the queried RR in the served zones

(2) Protocol errors such as SERVFAIL, FORMERR, NOTIMP

(3) nscount>0 and ancount=0 in the header

Table 3.7 Statistics for the QNAME length

	μ	σ	max
28-NOV-2003 (6249736 valid samples)	22.49	5.65	193
16-DEC-2003 (2997867 valid samples)	23.59	6.18	94

μ: mean value (bytes), σ: unbiased standard deviation (bytes)

max: maximum QNAME length (bytes)

Figure 3.4 shows the distribution of QNAME in the collected DNS answers of December 16, 2003. The CDF (Cumulative Distribution Function) shows that the percentage of DNS payloads whose QNAME length is equal to or larger than 43 bytes is less than 0.2% and very rare.

Figure 3.5 shows the distribution of the number of A RRs for each of the collected DNS answers of December 16, 2003. The author presumes that the reason why number of packets which the number of RR is 13 outstands is that the number of RRs for frequently-accessed zones such as the Root Zone and gTLD (generic Top Level Domain) zones (`.com`, `.net`, etc.) is 13.

The author also observed a payload with 75 RRs, which actually contained fragmented part of a payload longer than MTU (Maximum Transmission Unit). By performing the same query again by TCP later resulted in the unfragmented complete answer which has 150 RRs and 2665 bytes of payload length.

Some examples of payload length distributions for collected and simulated DNS answers

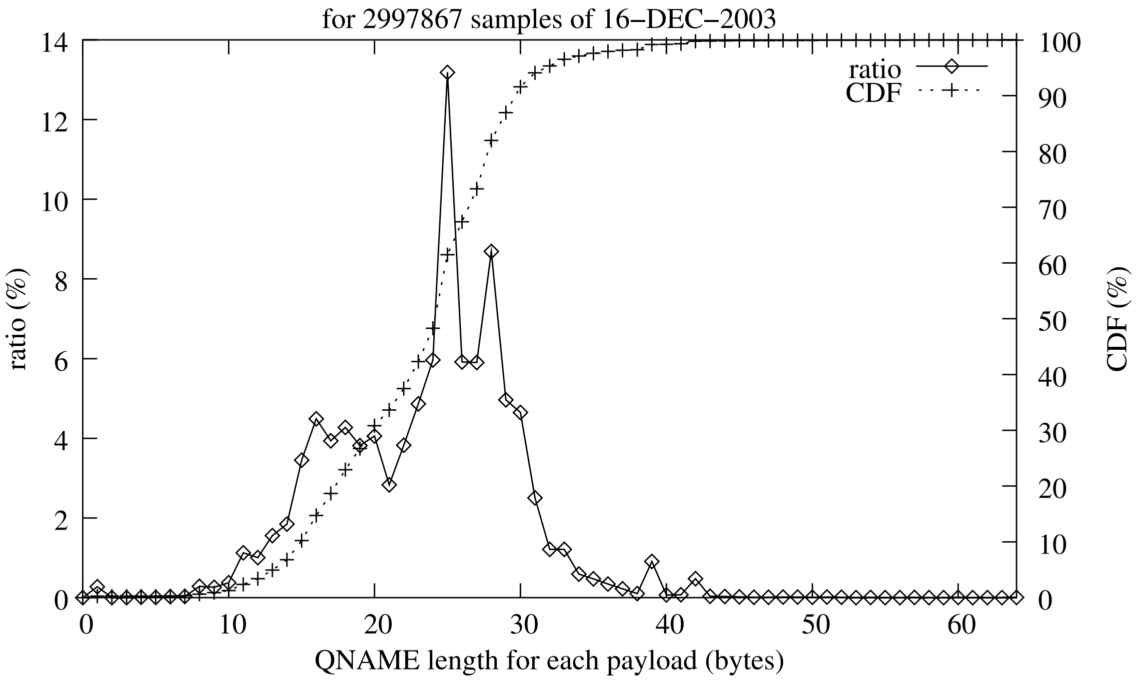

Figure 3.4 QNAME length and the CDF for each DNS answer of 2997867 samples collected on 16-DEC-2003

of December 16, 2003, are shown as follows; the raw collected data as Figure 3.6, the simulated data with an AAAA RR added for each existing A RR as Figure 3.7, and the simulated data with existing A RRs replaced by AAAA RRs as Figure 3.8. These figures show that the percentage of payloads larger than 512 bytes increases when AAAA RRs are added or replace the existing A RRs.

Figure 3.9 shows the CDFs of collected and simulated DNS answers, magnified to show the difference of the percentage of over-512-byte payloads before and after the simulations. While the percentage of over-512-byte payloads is less than 0.1% in the collected data, the percentage increases to $\simeq$2.77% for the simulation data of AAAA RRs added to A RRs, and to $\simeq$1.66% for the simulation data of AAAA RRs replaced A RRs.

3.5 Solutions for Handling Larger Payload Length

In this section, the author evaluates proposed solutions for handling larger DNS payloads and how effective they are.

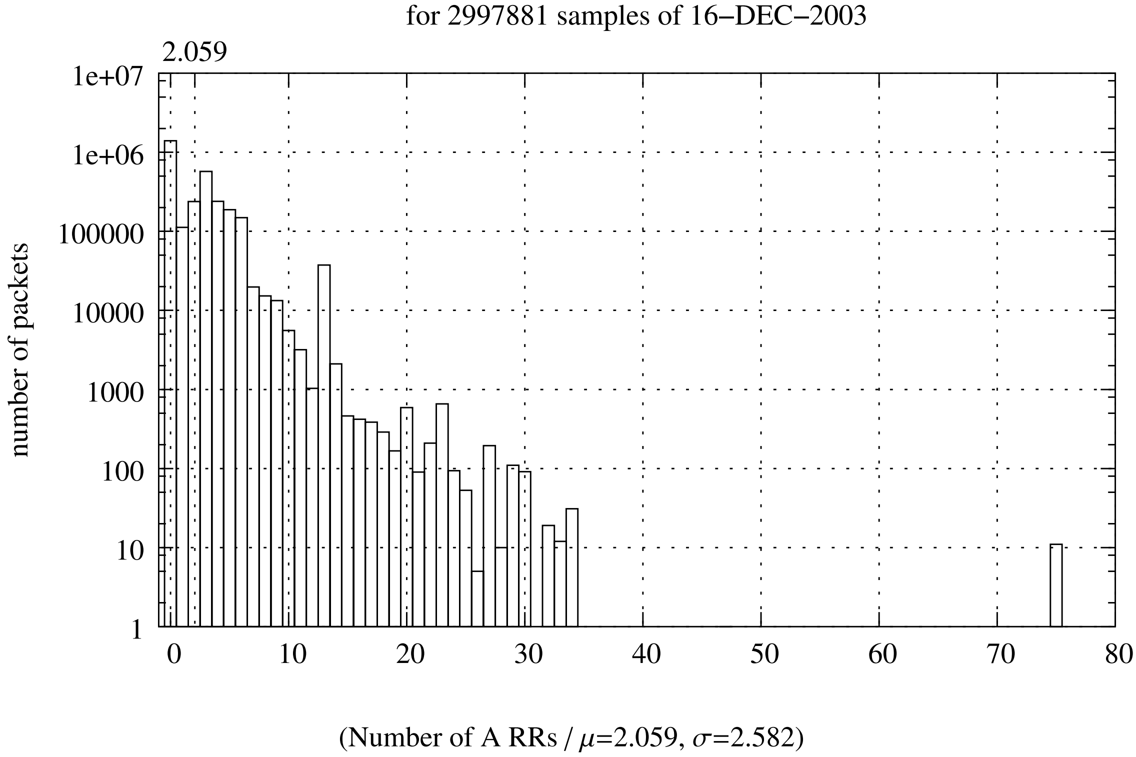

Figure 3.5 Numbers of A RRs for each DNS answer of 2997881 samples collected on 16-DEC-2003

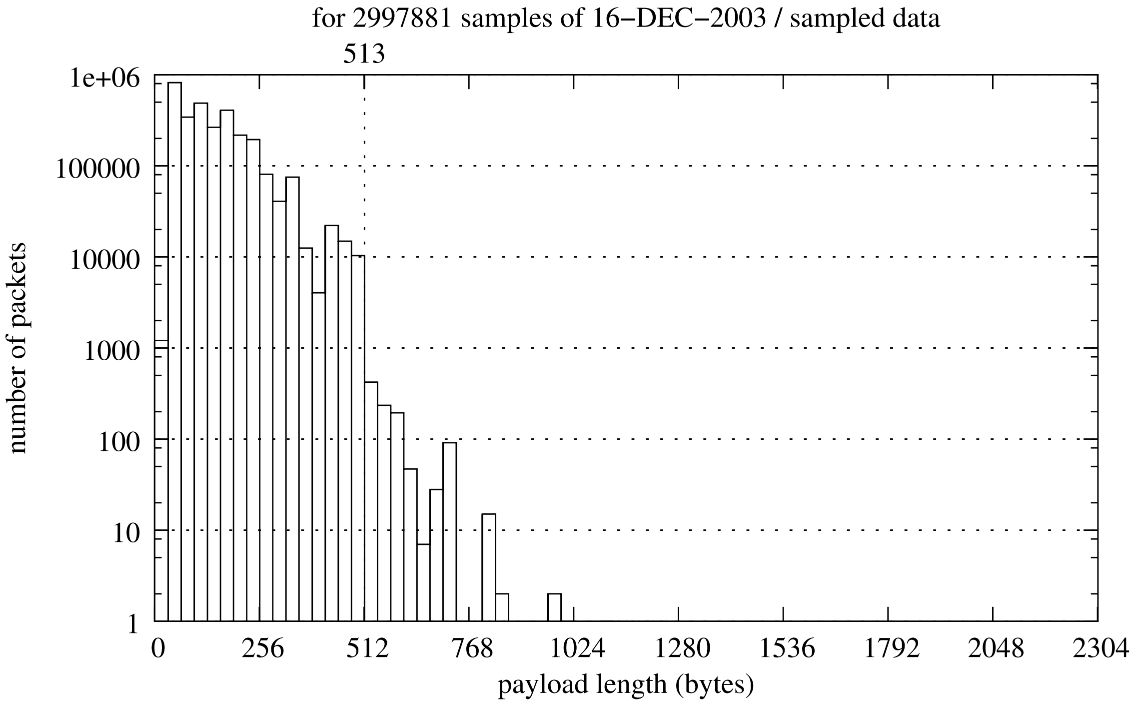

Figure 3.6 DNS answers of 2997881 samples collected on 16-DEC-2003

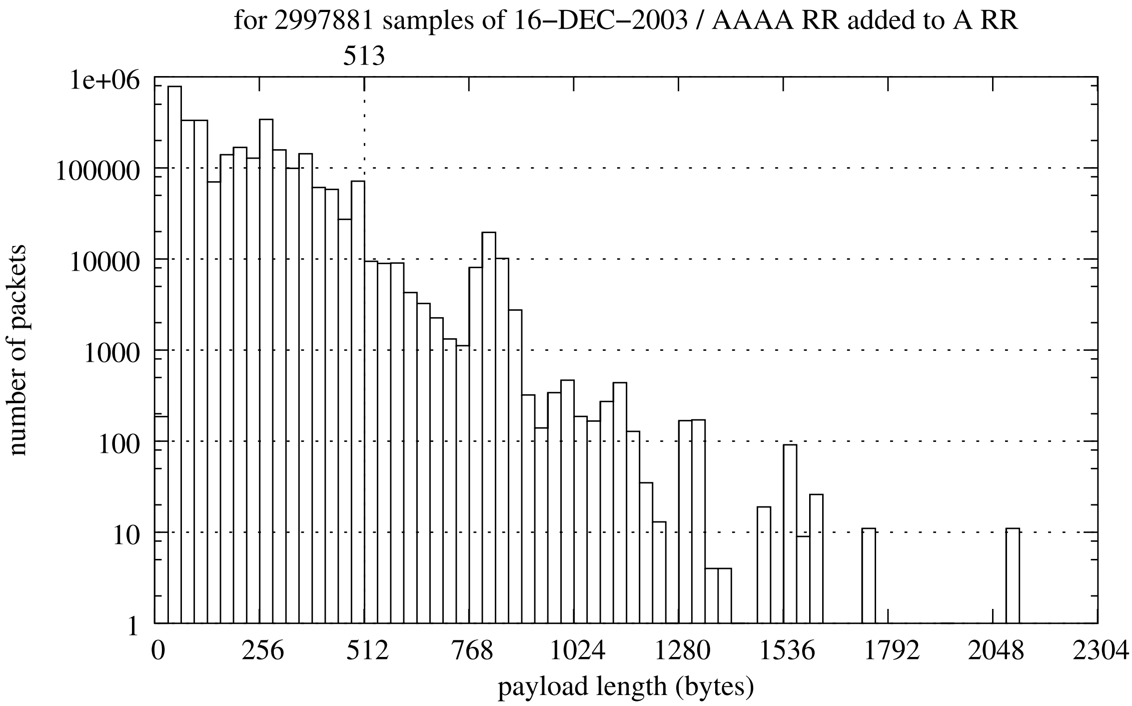

Figure 3.7 Result of a simulation adding an AAAA RR to each A RR for the DNS answers of 2997881 samples collected on 16-DEC-2003

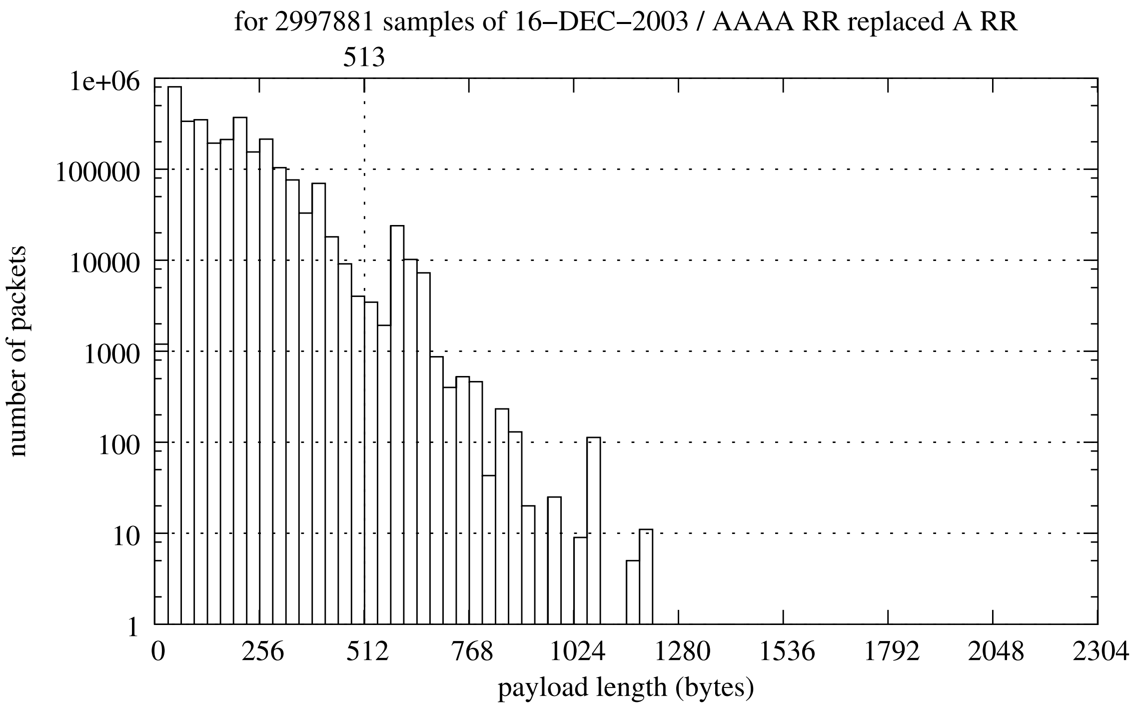

Figure 3.8 Result of a simulation replacing each A RR to an AAAA RR for the DNS answers of 2997881 samples collected on 16-DEC-2003

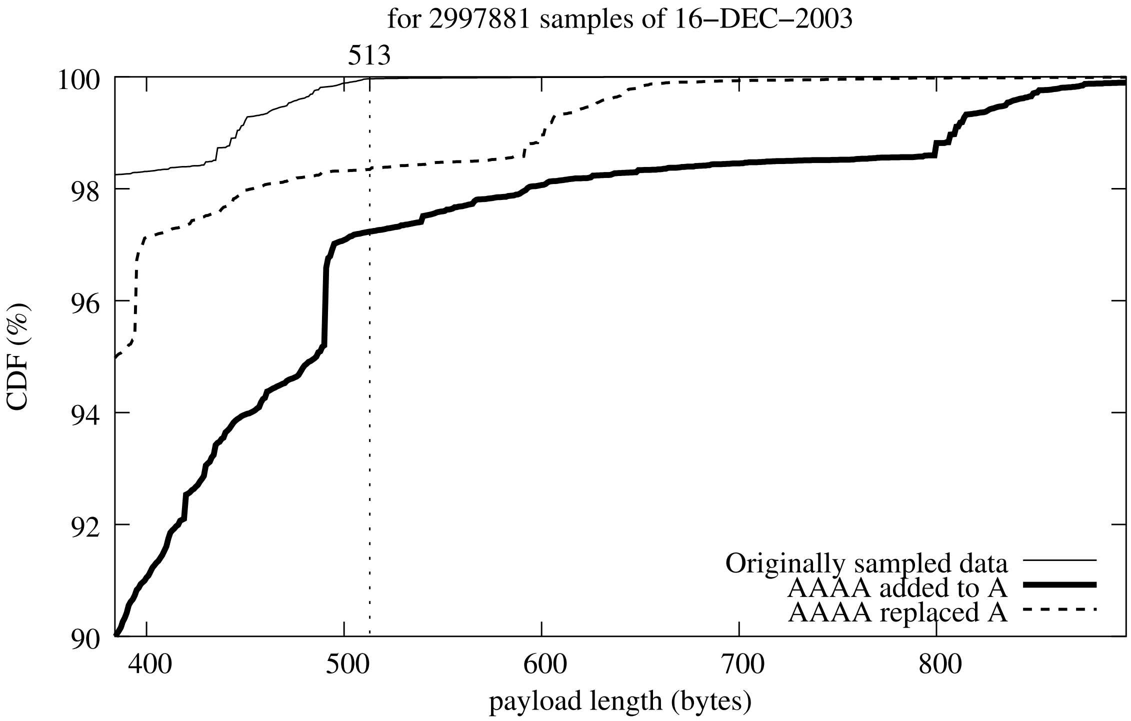

Figure 3.9 The CDF of DNS answers for 2997881 samples collected on 16-DEC-2003 and the simulation results

3.5.1 Prediction from Simulation Results

The result of the simulation in Section 3.4 shows that the percentage of DNS UDP payloads larger than 512 bytes increase from less than 0.04% to 1~3% during and after the migration from IPv4 to IPv6. This causes higher attempt rate of TCP retransmission of the larger payloads, and affects all DNS servers, so workarounds to reduce the impact should be established.

3.5.2 Payload Length Extension and Effects of EDNS0

An extension of DNS transport protocol called EDNS0 [6] is proposed to cope with the limitation of 512-byte maximum UDP payload length of DNS by RFC1035. EDNS0 defines a *pseudo RR* called OPT, which does not accurately fit into a definition of RR since it does not carry a pointer or a related data of a domain name. In this chapter, however, the author calls it as OPT RR since it has the data format of RRs.

On the EDNS0 protocol procedure, the servers and resolvers exchange the maximum length of UDP payloads which they can handle using the OPT RR so that they can exchange over-512-byte payloads.

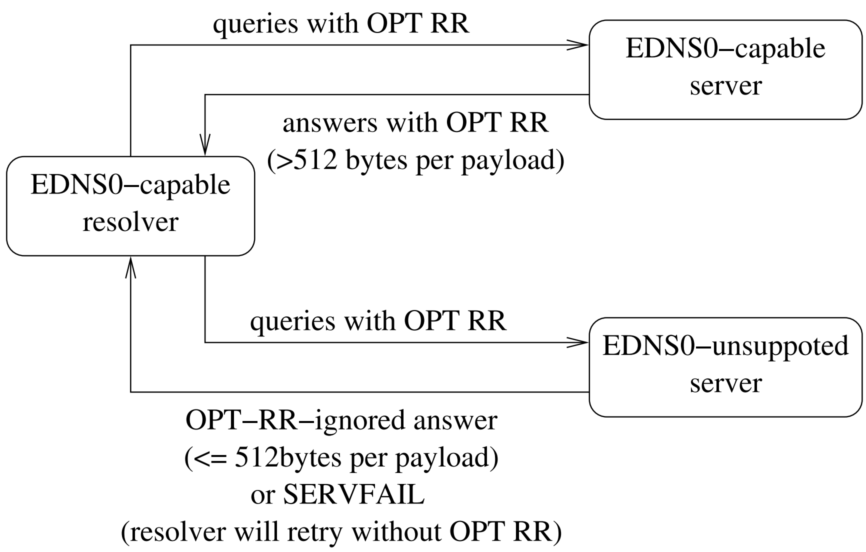

Figure 3.10 EDNS0 negotiation sequence between the servers and resolvers

The protocol negotiation procedure of EDNS0 is as follows, also shown in Figure 3.10:

- An EDNS0-capable resolver sends a query with OPT RR to tell the server the maximum payload length which the resolver can handle;
- If the server which receives the query is EDNS0-capable, then the server responds the answer with OPT RR to show that the server recognizes the proposed maximum payload length by the resolver;
- If the server is not EDNS0-capable, it either simply ignores the OPT RR, or returns an error code such as SERVFAIL;
- The resolver can find out whether the server is EDNS0-capable or not by checking the existence of OPT RR in the response or by receiving the error code from the server.

Table 3.8 shows the status of EDNS0 support among popular DNS servers. BIND [18], NSD [20], and Microsoft's Windows 2003 Server [73] are capable to handle EDNS0 and can configure the supported maximum payload length in the source code or by a runtime parameter. On the other hand, `djbdns` [19] does not support the EDNS0 extension.

Table 3.9 shows the usage details of EDNS0 and the specified maximum payload length from the collected DNS payloads. More than half of the whole answers are with OPT RRs showing the EDNS0 capability of the server. More than 2/3 of the answers have the UDPsize (acceptable maximum payload length) of 4096 bytes. According to these results, the author predicts that EDNS0 will become a popular DNS extension in the near future.

Table 3.8 EDNS0 support of popular DNS servers

Server name and the versions	EDNS0-capable? (yes/no)	maximum length (bytes)
BIND 8 and 9 (since 8.3.0)	yes	4096
djbdns-1.05	no	N/A
NSD 1 and 2	yes	4096
DNS Server of Windows Server 2003	yes	1280

(maximum payload length values shown are the default values, configurable in the source-code or by a runtime parameter)

Table 3.9 Usage details of EDNS0 and the specified payload length

for 3694918 answers of 28-NOV-2003 with OPT RRs (43.97% of 6249736 answers)				
UDPsize	512	1280	2048	4096
numbers	13	1399	706497	2987009
%	< 0.01	0.038	19.12	80.84
for 1318187 answers of 16-DEC-2003 with OPT RRs (59.12% of 2997881 answers)				
UDPsize	512	1280	2048	4096
numbers	1	1126	434632	882428
%	< 0.01	0.085	32.97	66.94

The author also considers that the small amount of payloads larger than 512 bytes shown in Figure 3.6 indicates that some DNS exchanges have already been extended with EDNS0.

3.5.3 The Overhead Imposed by EDNS0

Each OPT RR used for EDNS0 to transmit extended maximum payload length values consumes 11 bytes without any other extended fields. The size of OPT RR, however, affects very little to the overall payload length, provided that the length of other RRs are much larger than the OPT RR.

According to the measurement and simulation results shown in Table 3.5, the case which the payload length exceeds 4096 bytes rarely happens. The author claims that if all DNS UDP payloads could carry 4096 bytes as the maximum length by an EDNS0 extension, almost all the TCP retransmission of DNS payloads during or after the migration from IPv4 to IPv6 could be suppressed.

For implementing EDNS0 on DNS resolvers and servers, the memory area consumed for the data buffers of DNS payloads increases as the maximum payload length gets larger, but these buffer area will not become a significant processing overhead since they can be released immediately after the DNS query-and-answer transaction is complete.

For example, BIND version 9.2.3 has fixed-length buffer length for transmitting and receiving the DNS payloads regardless of EDNS0 extension. The internal data structure for EDNS0 has the total size of 60 bytes for the i386 architecture, which consists of the members of the structure `ns_client` called `udpsize` and `opt`, and the structure `dns_rdataset` used to store the OPT RR.

Assuming the internal data structure, if the server needed to keep 100000 simultaneous processing states, the memory area required for storing the states would be ≃6Mbytes, which can be stored without significant performance overhead on the modern computer hardware.

The UDP payload length increase enabled by EDNS0 will result in fragmentation of a UDP datagram to multiple IP packets due to the limitation of IP MTU length as shown in Figure 3.11.

IPv6 defines the minimum value of MTU to 1280 bytes (RFC2460 [74] Section 5). A unfragmented IPv6 packet of minimal configuration, with the IPv6 basic header (40 bytes as in RFC2460 Section 3) and the UDP header (8 bytes), can contain $1280 - (40 + 8) = 1232$ bytes of the UDP data payload. If the UDP payload of IPv6 exceeds 1232 bytes, each IPv6 packet must be sent with the *fragment header* attached by the host since on IPv6 the routers will simply discard the oversized packets without the fragment header.

The simulation results of AAAA RRs added to A RRs including ARs, which gives the largest payload length, shows that the percentage of payload length larger than 1232 bytes is ≃0.006% of the data based on the collected traffic of November 28, 2003, and is ≃0.018% of the data based on the collected traffic of December 16, 2003. Considering that ≃3% of the DNS answer payloads become larger than 512 bytes after performing the simulation, ≃99% of the UDP packets carrying the payloads larger than 512 bytes is predicted to be delivered without fragmentation.

3.5.4 TCP Overhead and the Improvement by T/TCP

DNS transport protocol falls back from UDP to TCP when the payload length exceeds 512 bytes. Due to the 3-way handshake procedure of TCP, a DNS exchange over TCP requires at

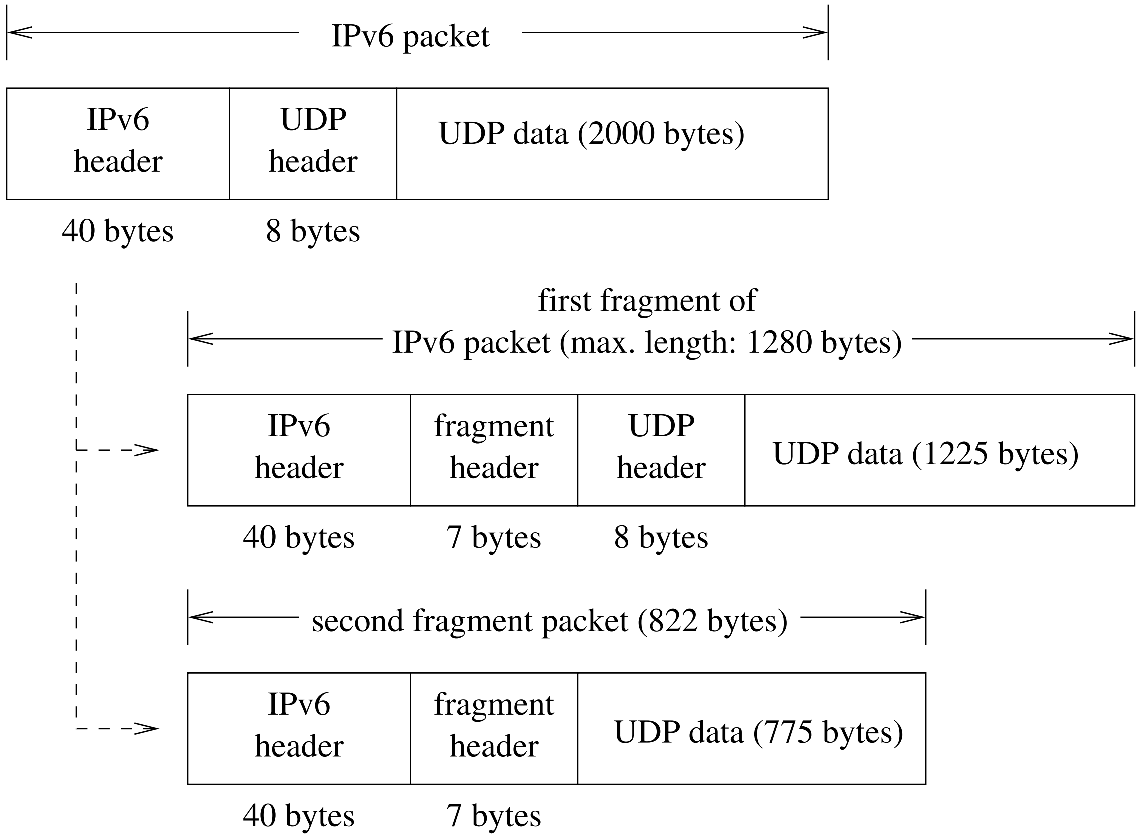

Figure 3.11 An example of IPv6 UDP fragmentation

least 5 packets. On UDP, only 2 packets are required instead.

Introducing T/TCP [21], a TCP protocol enhancement, reduces the number of packets for each exchange from 5 to 3 for the second and later exchanges between the same pair of resolver and server [75]. The traffic overhead of TCP fallback of DNS transport protocol can be reduced using T/TCP, as well as shortening the timeout period of disconnection to $\simeq 1/8$th of the traditional non-T/TCP timer value [21].

The features of T/TCP can be enabled by setting a run-time flag on FreeBSD, and can also be used on Linux [76]. The memory area overhead for T/TCP is minimal even if it exists, since T/TCP-capable operating system such as FreeBSD reserves T/TCP-specific memory structure for the TCP processing code.

T/TCP retains the reliability of TCP by the retransmission algorithm of TCP, and it has no requirement of application-level retransmission due to the fragmentation of the payload. Extensive use of TCP for TKEY RR exchange as shown in Table 3.3 is due to the requirement to reliable communication for key exchange of DNSSEC.

The author considers T/TCP is even more useful for exchanging DNSSEC payloads over

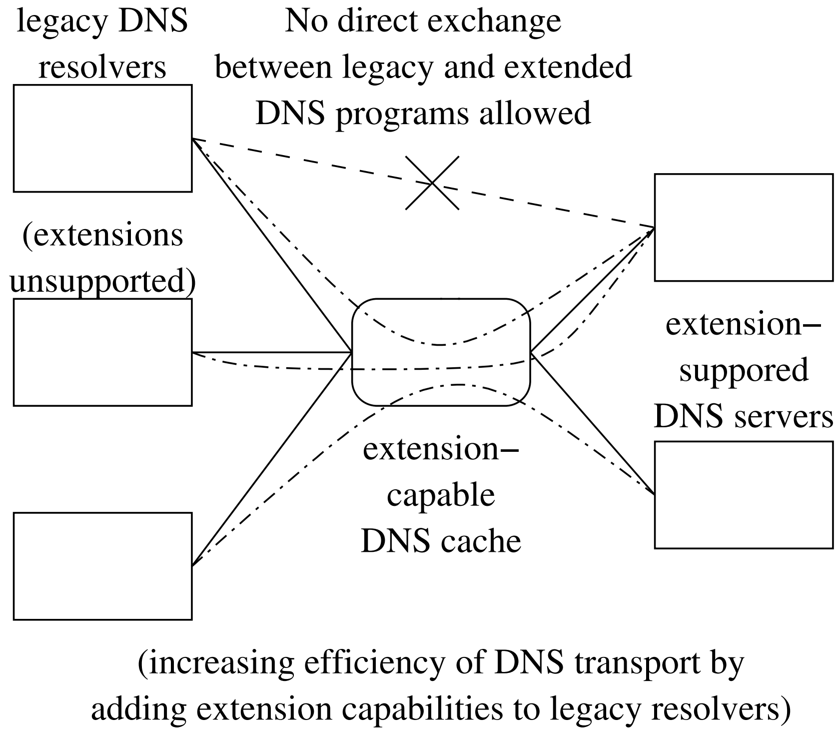

Figure 3.12 Increasing DNS transport efficiency by putting legacy resolvers behind an extension-capable cache

IPv6. Depending solely on EDNS0 and the UDP payloads may result in fragmentation of the UDP datagrams. RFC3226 Section 3 requires DNSSEC-capable servers and resolvers to be able to handle at least 1220 bytes of payload length as the minimum, and suggests being able to handle the payloads of 4000 bytes or more. Since the minimum guaranteed length of unfragmented UDP payload on IPv6 is 1232 bytes, only 13 more bytes than the minimum length are needed for a DNSSEC payload to cause IP-level fragmentation over IPv6, so the possibility of IP-level fragmentation of DNSSEC payloads over IPv6 is much higher than the non-DNSSEC payloads over IPv6.

The examples in DNSSEC's RFC2535 Section 5.4 show that 640-bit (80-byte) SIG (signature) RR is attached for each RR to be signed. The header for each SIG RR is at least 20 bytes even when the signer's name is compressed as described in RFC1035 Section 4.1.4, so the minimum length of each SIG RR is 100 bytes. Considering the example of the Root Zone, if the all 13 NS RRs representing the 13 Root Servers are signed, the corresponding SIG RRs consume 1300 bytes by themselves and have already exceeded the limit of maximum 1232 bytes for unfragmented UDP payload of IPv6.

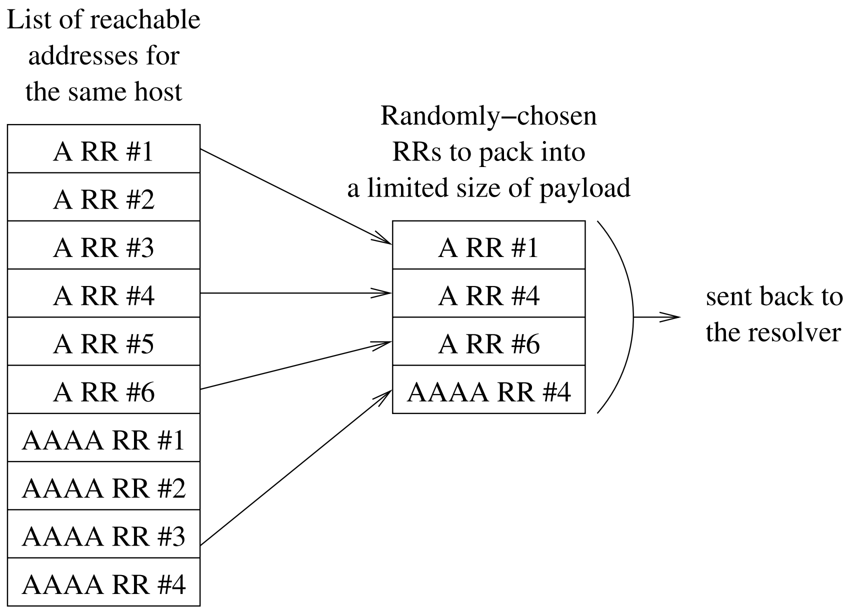

Figure 3.13 Choosing A and AAAA RRs ramdomly to fit them into a limited size of DNS payload

3.5.5 Using DNS Cache to Hide Extension-uncapable Servers and Resolvers

Most of the existing DNS resolvers and servers are not capable to handle protocol extensions of EDNS0 as in Section 3.5.2 and T/TCP as in Section 3.5.4. Those legacy resolvers and servers can be hidden behind a extension-capable DNS cache and use the cache as a protocol converter and relay program as shown in Figure 3.12. Mandating the usage of DNS cache for legacy DNS programs is a practical workaround for daily DNS operation to prevent the performance degradation caused by the DNS UDP payload length limitation.

3.5.6 Selecting IP Addresses to Answer for a Query

To prevent increasing of the payload length and avoiding protocol fallback of the current DNS UDP payload length limitation, the server selectively change the RRs to answer if multiple RRs are included.

For example, `tinydns`, a non-recursive DNS server program of `djbdns`, limits the number of A RRs in the answer to 8 and randomly choose the A RRs if the multiple A RRs should be answered for the queried name. For RRs representing IP addresses such as A and AAAA RRs, the host should respond in the same way whichever the IP address is used for a connection, so narrowing the choice of address-representing RRs will not cause an operational problem, as shown in Figure 3.13.

This sort of selective-answering strategy is effective to reduce the length of payload of DNS answers. Omitting the ARs specified in RFC2181 as described in Section 3.4 is another example of the same strategy to reduce the payload length by not sending the RRs which can be excluded from the answer RRs.

On the other hand, the selective-answering strategy is not applicable for RRs which cannot be omitted such as TXT RRs. The selective-answering of address-representing RRs is not unlimitedly applicable either from the DNS-operational point of view, since it is suggested that at least two or more address-representing RRs should be answered as ARs for an NS RR for stable DNS operation [72]. Durand, Ihren and Savola [50] also claim that omitting ARs based on the transport of the query would be problematic. So this selective-answering strategy is nothing more than an operating practice of DNS, and should *not* be considered as a permanent workaround.

3.5.7 Applying Selective-answering Strategy for IP Addresses of Root Zone

The selective-answering strategy of IP addresses explained in Section 3.5.6 is applicable to all DNS zones including the Root Zone. In this section the author discusses the condition and limitation of returning RRs for multiple servers for queries to the Root Zone. The author assumes that each Root Server has the one A RR and one AAAA RR at most, and that no exclusion of valid RRs occurs when answering the set of NS, A, and AAAA RRs for the same server.

The payload length of query for the SOA RR of Root Zone takes 12 bytes for the header, 5 bytes for the question section. In the authority section 75 bytes are needed for the SOA RR, including:

- the uncompressed domain name such as `a.root-servers.net`;
- 13 bytes for the first NS RR as the returned domain name of the authoritative server is compressed as a 2-byte index referring to that in the SOA RR; and
- 15 bytes each for the second and later NS RRs, whose partial name `root-servers.net` in the returned domain name is compressed as a 2-byte index, referring to that of a previous RR.

For the additional section, each A RR requires 16 bytes and an AAAA RR requires 28 bytes, providing that the domain name is compressed as those in the authority section.

Table 3.10 shows the maximum values of the number of servers within the limit of 512-

Table 3.10 The number of servers n within the limit of 512-byte payload length

Types of RRs returned	payload length	maximum n
A RRs only for IPv4-only networks	$90 + 31n$	13
pairs of A RR and AAAA RRs for IPv4+IPv6 networks	$90 + 59n$	7
AAAA RRs only for after-migrated-to-IPv6 networks	$90 + 43n$	9

byte payload length, calculated from the conditions previously mentioned.

The Root Servers are required to selectively choose the number of RRs included in the answer as in the list above. The processing load of the servers would not be fully balanced and certain specific servers would have more load, if the selective choice is biased. To balance the load, randomly choosing the servers is required.

Some popular implementations of DNS servers such as those in BIND and `djbdns` refer the 13 IPv4 addresses of the Root Servers as fixed values and load them in the start-up sequence. An issue arises to decide which set of values takes precedence over the other set, between the fixed values given at the start-up sequence and the values cached through DNS lookups. A possible solution is that the start-up fixed values are used when no Root Zone information is cached in the server program, and that the cached information takes precedence over the fixed values if it exists. By following this solution updated information from the Root Servers are used to look up the Root Zone even if the start-up fixed values are obsolete. BIND works as previously described.

Similar calculation can be performed for non-Root zones. QNAME length of the domain name of the zone should be more seriously considered since the maximum length of the QNAME is 257 bytes, while for the Root Zone the QNAME length is only 1 byte. JPRS [72] calculates the requirement for the authoritative servers of `jp` domain, and the result for the maximum number of the authoritative servers which can be included in a 512-byte-maximum UDP payload with both A and AAAA RRs, is 3. This indicates that the maximum number of RRs which can be answered in a single payload is affected by the length of the name in a DNS zone.

Vixie and Kato [77] have performed a simulation for estimating how many AAAA RRs could be added to the current set of Root Servers with 13 names under a common parent domain `root-servers.net`, and 13 A RRs for the IPv4 addresses. They conclude that adding 2 to 5 IPv6 address (AAAA) RRs would not have a significant negative operational impact on the domain name system, by allowing selective response in the ARs, which consists of the A and AAAA RRs.

3.6 Concluding Remarks

In this chapter, the author has presented that the percentage of DNS answers exceeding the 512-byte UDP payload size limit, including the additional records, increases from 0.04% to 1~3% during and after the migration from IPv4 to IPv6 with a simulation by packet-length recalculation, using the real-world DNS traffic data taken from a large-scale campus network.

The author has also presented the effectiveness of EDNS0, a DNS protocol enhancement to prevent performance degradation by protocol fallback to TCP, and setting the maximum payload length to 4096 bytes by EDNS0 effectively prevents the DNS protocol fallback to TCP after the migration to IPv6 completes.

The author also discussed the effectiveness of T/TCP, a TCP enhancement, to prevent performance degradation by increase of number of packets exchanged during the resolver-server exchanges over TCP, and the usefulness of the T/TCP for DNSSEC-based resolver-server exchanges over IPv6.

Limiting the payload length by selectively choosing the RRs to answer to a DNS query was also discussed, although the author claimed that the method should not be considered as a solution and rather treated be a practical workaround to cope with the current 512-byte limitation of DNS UDP payload length.

The author proposed the following three action items should be realized as soon as possible over the whole Internet to prevent unnecessary fallback to TCP of DNS transport protocol and to reduce the DNS server workload:

1. promoting the EDNS0 and making the maximum length practically allowed for DNS UDP payload large enough to accept further increase of DNS payloads and the RR length;
2. hiding the legacy non-EDNS0 resolvers and servers behind EDNS0-capable servers and caches to reduce the possibility of TCP fallback of DNS transport protocol; and
3. introducing T/TCP to improve the efficiency of TCP exchange itself for DNS.

The further research issues to be resolved include the measurement of DNS traffics on Root Servers and large-scale Internet service providers [78, 79], and the measurement and estimation of computational overhead and network workload of EDNS0 and T/TCP.

Chapter 4

T/TCP for DNS: A Performance and Security Analysis

4.1 Introduction

Internet systems nowadays are always under continuous and persistent attacks, as the social and business activities become dependent on Internet. All Internet services are the targets of the intruders to exploit. DNS is of no exception. DNS security is critical for the stability of Internet.

Reliable communication transport for DNS is essential to establish an authenticated DNS exchange by DNSSEC. While extending the UDP payload length by EDNS0 is effective for reducing the overhead imposed by the larger payloads generated by DNSSEC, improving the efficiency of TCP transport for DNS exchange is also essential to reliably exchange authenticated RRs.

In this chapter, the author focuses on the transport security issues from an administrative point of view, and proposes an alternative DNS transport with T/TCP (Transactional TCP) [80, 21], for improving the *overall* security of the DNS. The author describes how T/TCP helps ensuring the transport security of DNS, by showing the practicality of replacing the current DNS queries of UDP with T/TCP, through the performance analysis and evaluation. As T/TCP is an extension of TCP, it preserves many advantages of TCP to UDP, such as reliable error-free data exchange, sophisticated retransmission algorithm, and the more detailed traffic controllability on the firewalls.

In the later sections, the author describes the T/TCP fundamentals and the advantages to traditional TCP in Section 4.2, and the evaluation results of T/TCP used as a DNS transport in Section 4.3. The author concludes this chapter on Section 4.4 with a discussion of the possible application fields of T/TCP to improve DNS reliability and the security.

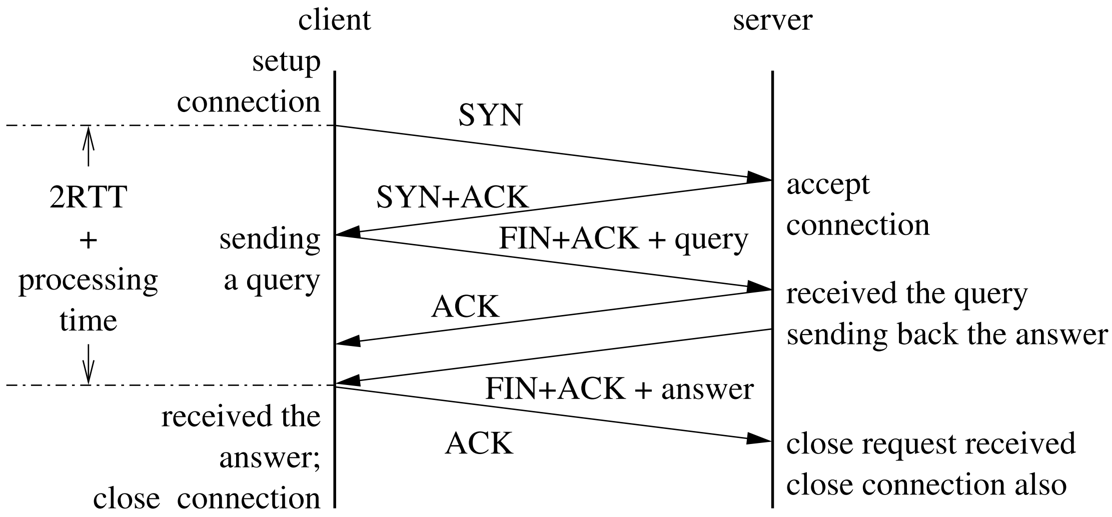

Figure 4.1 Traditional TCP time line

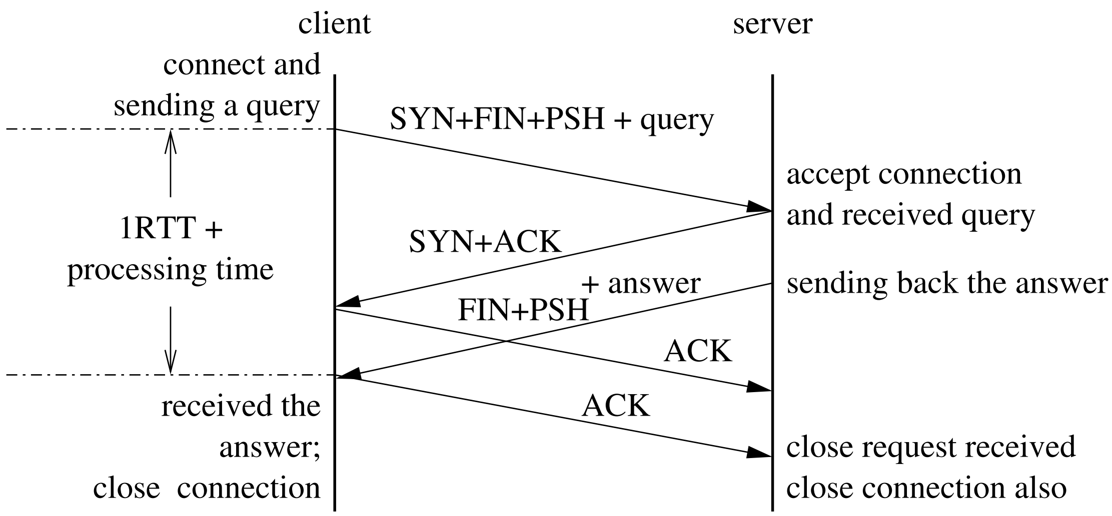

Figure 4.2 T/TCP time line

4.2 T/TCP and Traditional TCP

In this section, the author describes the fundamentals of T/TCP and how it differs from the traditional transport protocols, UDP and TCP.

T/TCP is an extension of TCP. The concept model of T/TCP [80] was proposed in 1992 and later updated by the functional specification [21] in 1994. Stevens [81] gives a detailed analysis of T/TCP as a book chapter. As of March 2003, FreeBSD and Linux operating systems have T/TCP-compatible kernels.

4.2.1 T/TCP Communication Model

T/TCP is designed for a transactional use between a connection-based client-server communication, which proceeds as the following sequence:

- The client sends a request to the server;
- Then the server sends back the reply;
- The exchange completes and the link is disconnected.

Some of the suggested applications of T/TCP include HTTP (Hypertext Transfer Protocol), RPC (Remote Procedure Call), and DNS queries [81].

In this chapter, however, the word *transactional* is solely limited to explain the communication attribute which T/TCP assumes. The readers are advised that the word *transactional* does not mean that the T/TCP itself meets the all requirements of database transactions. For example, a T/TCP exchange could be duplicated in a certain extreme case such as when the server host crashes and reboots before sending back the reply for a request from a client. Also, the protocol does neither support the rollback operation, which is required for the two-phase commit sequence, nor guarantees the atomicity of the client-server exchange. Those database-specific properties should be performed by the application programs and protocols, and are out of the scope of this dissertation, since DNS database queries allow duplicate answers.

4.2.2 T/TCP and TCP Time Lines

Using traditional (non-transactional) TCP for the transactional model of Section 4.2.1 sequence requires two round-trip exchanges. Figure 4.1 shows the time line of traditional TCP. It shows that the first of the two exchanges is solely for setting up a TCP connection, while the second one is actually used for the data exchange.

On the other hand, using T/TCP requires only one round-trip exchange, which is the same as in the UDP case. Figure 4.2 shows the time line of T/TCP. It shows that the first packet sent from the client to the server carries the query data as well as the connection request. Putting the query data on the same packet for the connection request is performed by using the CC (Connection Count) options of TCP, introduced by T/TCP, to indicate the support and to avoid duplicate old connections, as described in Section 4.2.3.

4.2.3 TAO Test

A TCP server needs to find out whether a received packet with the `SYN` flag set really means a new connection. Traditionally this is performed by performing the three-way handshake shown in Figure 4.1, as the client and server acknowledge `SYN` request with each other.

On T/TCP, a mechanism called TAO (TCP Accelerated Open) is introduced to allow a T/TCP server to know that a `SYN` request from a T/TCP client is new, without the three-way handshake. An identifier called *connection count* (CC), a 32-bit integer, is assigned to each connection that a host establishes. The CC cache is maintained per each peer host.

Three new TCP options, `CC`, `CCnew` and `CCecho`, are defined for T/TCP as follows:

- The `CC` option carries the CC value in an initial `SYN` segment of the T/TCP client, or in the other segments if the other end sent a `CC` or a `CCnew` option with `SYN`.
- The `CCnew` option only appears in an initial `SYN` segment, when the client needs to perform the traditional three-way handshake while indicating the support of T/TCP.
- The `CCecho` option only appears in the `SYN+ACK` segment of a three-way handshake (from a T/TCP server), and echoes the received connection count value of a `CC` or `CCnew` option to tell that the server understands T/TCP.

Each T/TCP host performs the following procedure, called the TAO test, to decide whether to use TAO or not when a `SYN` request is received:

- When no cached value of CC is found for a peer host or a `CCnew` option is received, a three-way handshake is performed with the CC options and the CC values are exchanged, and the CC cache for the peer host is initialized.
- If a CC value is cached for a peer host, verification of a CC option in the received packet is performed:
  - If no CC option is found, the CC cache is cleared and the connection falls back to the three-way handshake sequence.
  - If a CC option is found with the received packet, verification for the value of received connection count by comparing it to the cached value. If the received value is greater, the `SYN` is recognized as a new one and accepted without the three-way handshake. If not, the CC cache is cleared and the connection falls back to the three-way handshake sequence.

When the TAO test fails, the data payload carried with the initial `SYN` request of the T/TCP is not passed to the application software. By performing the TAO test, T/TCP can avoid duplicate old connections without performing the three-way handshake every time.

T/TCP has the overhead for each pair of connected hosts to initialize the per-host CC cache of both on the client and the server. This initialization is, however, only required to perform for the first transaction between the two. Once the CC cache is properly initialized, the client and server pair will use the accelerated handshake sequence for the second and the later transactions, as long as the CC update is properly continued without external interference such as an intrusion attack of a spoofed host.

T/TCP CC cache consumes some amount of memory, though it is predictable and does not impact the system performance unless the available memory space for the kernel is limited. For example, on FreeBSD 4.7-RELEASE, the T/TCP-specific memory resources are listed as follows:

- A kernel 4-byte counter `tcp_ccgen` is allocated for each kernel to give CC values per each connection;
- For each host, two 4-byte variables called `tao_cc`, `tao_ccsent`, and a 2-byte variable called `tao_mssopt`, total 10 bytes, are allocated, as a per-host cache;
- For each TCP connection, three 4-byte variables called `cc_send`, `cc_rcvd`, and `t_starttime`, total 12 bytes, are allocated, as a part of the TCP control block.

For example, when 10000 hosts and 100 simultaneous T/TCP connections per each host are connected (1000000 connections total), the total number of bytes consumed is $(4 + 10 \times 10000 + 12 \times 100 \times 10000) = 12100004$ bytes. The memory block of this size is practically affordable for the PC servers which has usually a few hundred megabytes of the main memory installed.

4.2.4 DoS Immunity

T/TCP has some immunity against simple DoS attacks which UDP does not, by performing the TAO test for each transaction. Here are some scenarios:

- For example, in case of a simple DoS attack of multiply sending the same packet, UDP has no mechanism of rejection. On the other hand, when using T/TCP, the TAO test fails from the second and later received packets, as it mandates that the CC value must monotonically increase for each transaction. The failure of TAO test leads into the protocol fallback to the traditional three-way handshake procedure. Without the completion of the handshake, the data payload in a transaction request packet will not be transferred to the application software.

- In case of a distributed DoS attack, meeting the requirement of monotonic increase of the CC value for each transaction at the server for a successful attack is highly improbable unless the sequence of received packets from the attacking hosts is thoroughly controlled.
- When the attackers use an spoofed source address of IP packets to anonymize themselves, the first CC initialization sequence of the TAO test will not be completed, and the data payload will not be accepted. If the host specified by the spoofed source address exists, the host sends an `RST` packet as the reply for a non-existent connection.

These examples show that the T/TCP does not have a weakness of UDP which blindly accepts all incoming packets. While T/TCP does not authenticate the data payload itself and may exchange a larger number of packets than UDP does in case of a successful DoS attack, the protection of the TAO test gives an advantage to T/TCP from UDP against a DoS attack.

Some of the firewall products have employed the stateful analysis and inspection of traffic, which means the firewall internally verifies the protocol sequence of TCP. Enhancing this to verify the CC counts would be helpful for handling T/TCP through a firewall.

4.2.5 TIME_WAIT State

T/TCP has another feature to shorten the time spent in TCP `TIME_WAIT` state which is to complete full-duplex closing of a connection and to allow old duplicate TCP segments to expire.

The amount of time spent in the `TIME_WAIT` is traditionally specified as twice the MSL (Maximum Segment Lifetime). On FreeBSD 4.6.2-RELEASE and the 4.7-RELEASE, the default MSL is 30 seconds, so the length of `TIME_WAIT` state for the traditional TCP is 60 seconds.

On the other hand, T/TCP specifies the length of `TIME_WAIT` as eight times the RTO (Retransmission Timeout) when the connection duration is less than the MSL. RTO is a dynamic value estimated using the measured round-trip time on the network link with a pre-defined minimum value. For example, the minimum RTO estimated by FreeBSD 4.6.2-RELEASE and the 4.7-RELEASE is 1 second. So the length of `TIME_WAIT` state of T/TCP is shortened approximately to 8 seconds, when the actual RTT of the link is much smaller than 1 second.

A smaller length of `TIME_WAIT` state means a smaller size requirement to the network control block, and an increase of number of TCP connections which a server host can simultaneously handle.

4.2.6 Backward Compatibility

As T/TCP is an extension of TCP, it is backward-compatible with the traditional TCP. When the server is T/TCP-aware, it can identify the client is T/TCP-aware or not, since a T/TCP-aware

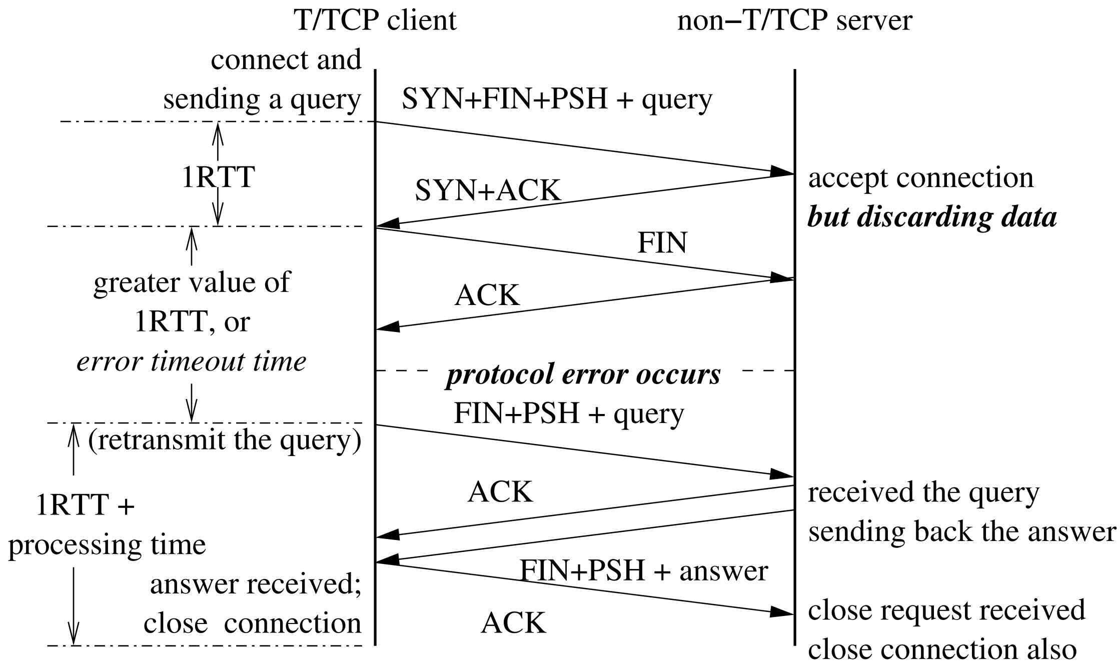

Figure 4.3 Time line of T/TCP client and non-T/TCP server on FreeBSD 4.6.2-RELEASE

client will send a TCP connection request with a CC option, while a traditional TCP client does not. Figure 4.1 applies in the case of a traditional TCP client and a T/TCP-aware server.

A fallback procedure must be followed in case of a T/TCP-aware client and a traditional non-T/TCP server. Figure 4.3 shows the procedure and the time line of FreeBSD 4.6.2-RELEASE. In this case, the `SYN` cache [82] of the server discards the data payload on the first packet, to avoid TCP `SYN`-flooding, a popular DoS attack which intends to consume the memory area allocated for the network control blocks. The server does not recognize the CC options either, so the first packet the client sends is treated only as a connection request. While this behavior is practically acceptable to protect the server from the possible `SYN`-flood attacks, it has an adverse effect of forcing the client to wait for an additional error timeout period for each transaction. Nevertheless, the backward compatibility of T/TCP to the traditional TCP is still retained.

Note that in either time line figures of Figures 4.1 or 4.2, the meaning of the `ACK` bit in TCP header is left unchanged. The packet filtering rules of allowing only *established* connections of TCP are applicable to T/TCP with no need to change.

4.2.7 T/TCP Programming

Modifying existing network programs to be T/TCP-compatible is a straightforward task, since the protocol details are all implemented in the kernel of the operating system. For example, on BSD-derived operating systems, a flag in the include file `<sys/socket.h>` contains the flag `MSG_EOF` to show the T/TCP support. An example of the necessary changes in FreeBSD [83] is as follows:

- On the server side, using `setsockopt()` system call for adding `TCP_NOPUSH` option to the listening socket is required to avoid unnecessary fragmentation of TCP segments.
- On the client side, the `connect()`-`write()`-and-`shutdown()` flow of system calls to initiate TCP connection and sending the query data must be replaced by a `sendto()` system call with `MSG_EOF` flag, since the T/TCP connection is implicitly established by the `sendto()` system call. The `TCP_NOPUSH` socket option is required as well.

To enable or disable the T/TCP functionality of a FreeBSD host, the administrator sets the kernel MIB (Management Information Base) variable of `net.inet.tcp.rfc1644` to 1 or 0, respectively. This value can be dynamically changed without rebooting the host.

4.2.8 Migration Issues

A few migration issues as follows should be considered on using T/TCP:

- The default state of T/TCP functionality is disabled in FreeBSD 4.6.2-RELEASE, as the document [21] is still considered *experimental* in the IETF and the standardization process.
- The system administrators must be aware that all TCP-related security attacks are also applicable to T/TCP.
- Some systems with a high security concern is configured to simply ignore the TCP packets with the `SYN+FIN` flags to avoid revealing the protocol stack of the operating system. In this case, T/TCP packets do not get through.
- The migration should begin with the server-side first, to avoid the error-timeout issue described in Section 4.2.6.

The following is the perspective of the author to these migration issues:

- The reason that IETF status of T/TCP is *experimental* is that the usage is limited to a single-query-and-single-answer transactional application. DNS database query will largely benefit from T/TCP especially when the query result no longer fits into a UDP packet because of increasing IPv6 address usage. The author believes some actual deployment of T/TCP for DNS is essential, since T/TCP has already been implemented and ready to be used.

- As T/TCP is an extension of TCP, T/TCP is also prone to the security attacks to TCP. The author considers, however, that the security risk imposed by the introduction of T/TCP is minimized by a proper security protection such as the TAO test.
- When a system rejects all the `SYN+FIN` packets, no T/TCP connection request and the reply can be used to communicate with the system. Avoiding usage of T/TCP is a practical workaround for such a system.
- The reason the author suggests to migrate first from the servers is that the programming needed for the migration is small, such as by enabling the TCP socket option of `TCP_NOPUSH` on FreeBSD.

4.2.9 What T/TCP Provides for DNS

The author proposes T/TCP as a replacement of the existing DNS UDP transport. The author considers that the migration from UDP to T/TCP is feasible by the following reasons:

- T/TCP has the immunity against DoS attacks by the TAO test, as described in Section 4.2.3.
- T/TCP is backward-compatible with TCP as described in Section 4.2.6. This ensures the connectivity during the migration phase, when T/TCP and TCP DNS hosts coexist.
- T/TCP has already been implemented in the production-level server operating systems such as FreeBSD and Linux, so for these systems the migration cost is small. Using these systems as DNS caches is a practical workaround for non-T/TCP systems, which are mostly running resolvers only.
- The programming cost for migration of a TCP program is small, as described in Section 4.2.7. The author needed less than 100 source code lines to modify `djbdns` [19] to make it T/TCP-compatible. The protocol stack implementations of T/TCP can be obtained as free software such as FreeBSD and Linux, and the detailed reference is available as a book [81].

```
/*
 * example quoted from function socket_send4()
 * of socket_send.c in djbdns-1.05 by Daniel J. Bernstein
 * modified by Kenji Rikitake
 */

/* sendto() system call for UDP */
sendto(s,buf,len,0,(struct sockaddr *) &sa,sizeof sa);

/* sendto() system call adapted for T/TCP
   with TCP_NOPUSH and MSG_EOF flags for T/TCP */
setsockopt(s,IPPROTO_TCP,TCP_NOPUSH,&opt,sizeof opt);
sendto(s,buf,len,MSG_EOF,(struct sockaddr *) &sa,sizeof sa);
```

Figure 4.4 Comparison of `sendto()` system call for UDP and T/TCP, for writing a T/TCP client (resolver for DNS) program in C

4.3 Evaluation of T/TCP

In our research, the author tested T/TCP as a DNS transport by modifying the program code of `djbdns` and measuring the performance and behavior. In this section, the author describes the details and the results of the performed experiments.

4.3.1 Test Environment

The software packages chosen for the experiment are listed as follows:

- FreeBSD 4.6.2-RELEASE and the 4.7-RELEASE as the operating systems, for the stability of the T/TCP implementations:
- `djbdns` as the DNS software, for the highly-modularized structure;
- `dummynet` [84], for simulating random packet loss and high-latency links.

The modification details of `djbdns` for the T/TCP support are listed as follows:

- adding a function to set the `TCP_NOPUSH` socket flag, and an interface to `sendto()` system call for `djbdns` socket library (examples shown in Figures 4.4 and 4.5 [19],

```
/*
 * example quoted from function socket_bind4_reuse()
 * of socket_bind.c in djbdns-1.05 by Daniel J. Bernstein
 * modified by Kenji Rikitake
 */

/* the variable s is passed to bind() system call */
int opt = 1; /* opt is used for enabling an option */
/* option for enabling the local address reuse */
setsockopt(s,SOL_SOCKET,SO_REUSEADDR,&opt,sizeof opt);
#ifdef MSG_EOF /* T/TCP */
/* adding TCP_NOPUSH option to ensure the T/TCP usage */
setsockopt(s,IPPROTO_TCP,TCP_NOPUSH,&opt,sizeof opt);
#endif /* MSG_EOF */
```

Figure 4.5 modification of `bind()` system call using `setsockopt()` for UDP and T/TCP, for writing a T/TCP server program in C

including the minimal modification for the client (resolver for DNS) and server code);

- changing the DNS resolver interface functions called from the `djbdns` programs to use T/TCP instead of traditional TCP; and
- changing `dnscache`, the DNS cache program, to use T/TCP for accepting the connections and external lookups.

The conditions of DNS query time measurements are as follows:

- `dns_resolve()`, a DNS resolver function of `djbdns`, is called for each query. A modified version is used to perform TCP-only DNS queries. `DNSCACHEIP`, The environment variable is set to choose the appropriate `dnscache` to test.
- Each query contains a request to the NS RRs of the Root Domain (`"."`), which `dnscache` can answer solely by referring to a configuration file `root/IP/@`, with no external or internal lookup.
- Choosing the T/TCP or traditional TCP is done as explained in Section 4.2.7.

Table 4.1 Total elapsed time of 1000 sequential DNS queries to a `dnscache` server (in seconds)

	local	Ether	ADSL
RTT (ms)	≈0.04	≈0.4	60~70
UDP	0.22	2.40	67.77
T/TCP	0.52	8.70	74.70
TCP	0.53	8.92	138.80

RTT: Round-Trip Time

4.3.2 The Protocol Overhead

Table 4.1 shows the result of measuring the difference of query processing time between UDP, T/TCP and TCP for different types of links. The author used a local interface, a 100BASE-TX Ethernet, and an ADSL (Asynchronous Digital Subscriber Link) of an Internet service provider.

For the local interface and Ethernet links, UDP is the fastest, since the number of packets exchanged for each query differs; 2, 5, and 6 for UDP, T/TCP, and TCP, respectively. On the other hand, the testing of the ADSL link shows that the overhead of T/TCP to UDP is only 10% of the total time, while TCP takes about twice as much as UDP does. This is consistent with the time line explanation on Section 4.2.2, as in the ADSL case, the RTT (Round-Trip Time) is much larger than the query processing time, and becomes a major portion of the total elapsed time.

4.3.3 On Allocated Connection Blocks

The author performed a test on how the number of allocated sockets (connection blocks) changes between TCP and T/TCP. The author performed 10000 queries of each transport protocol by 10 concurrent processes of 1000 sequential queries (total 10000) each, and measured how the number of active connection blocks from the beginning of the queries. The author evaluated how the `TIME_WAIT` value affects to the total processing time of the simultaneous query connections. The host used for this test has only ≈8000 connection blocks available to the cache program, acting as a DNS server. The server and the clients were connected through the local interface.

Figure 4.6 shows the result. In the beginning, the number of the allocated socket increased at the rate of ≈1900 queries/sec, but after the connection blocks were used up by the query-generation processes, they waited until the first `TIME_WAIT` period expires; the suspended queries were processed later as the connection blocks became free after the `TIME_WAIT` state

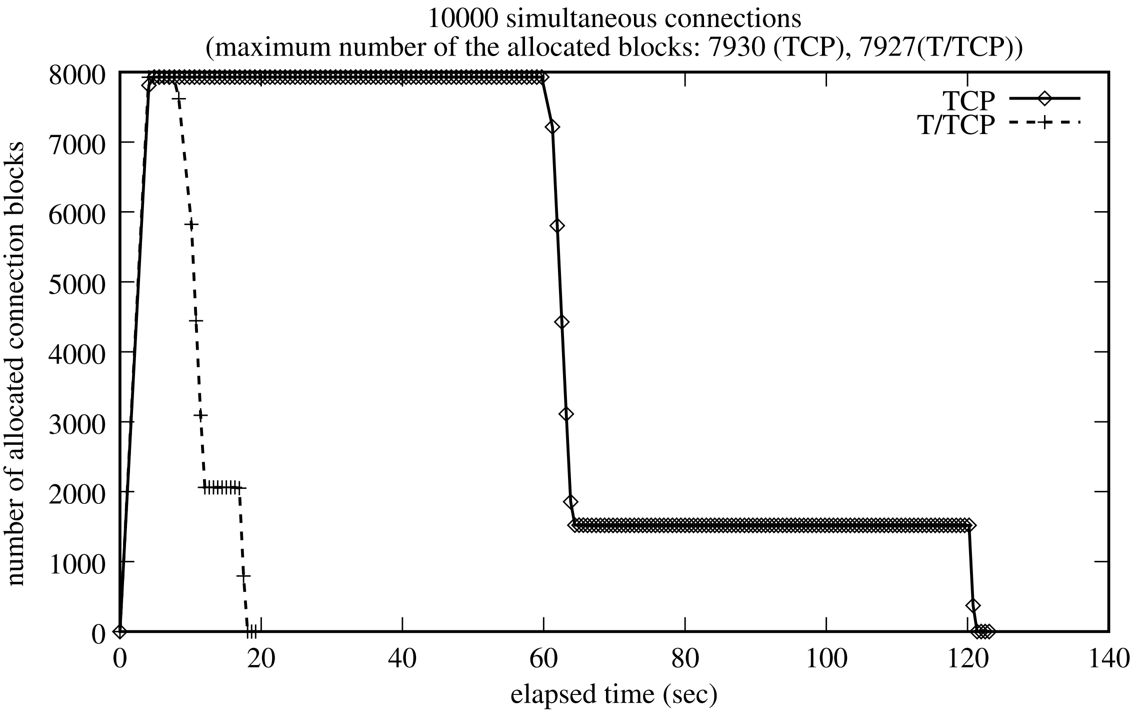

Figure 4.6 How the `TIME_WAIT` value affects the number of allocated connection blocks for 10000 simultaneous connections

completion. For an application which accepts a large amount of queries, using T/TCP instead of TCP will reduce the total waiting time of queries to approximately two-fifteenth (8RTO / 2MSL $\simeq$ 8 / 60), which is shown in Figure 4.6 as the length of time from the beginning of the test to when the number of allocated connection blocks starts falling from the largest value ($\approx$8 seconds on T/TCP, 60 seconds on TCP). This behavior is consistent with the explanation on Section 4.2.5, which suggests the length of the `TIME_WAIT` value shortened from 60 seconds to $\approx$8 seconds by the protocol change from the traditional TCP to T/TCP.

4.3.4 On Packet Loss Rates

The author performed a test to evaluate how the random packet loss rate affects the query success rates of UDP and T/TCP. Since UDP exchange takes 59 seconds as the maximum value by the retransmission algorithm in Section 2.2.4, the value of T/TCP timeout to determine the success of query is extended from the default value of 10 to 60 seconds on both the server and the resolver sides. The author used two hosts connected with a 100BASE-TX link and `dummynet` for simulation. 1000 concurrent queries were conducted for each random packet loss rate value. Two delay cases, none and 500 milliseconds for simulating mobile access environment were

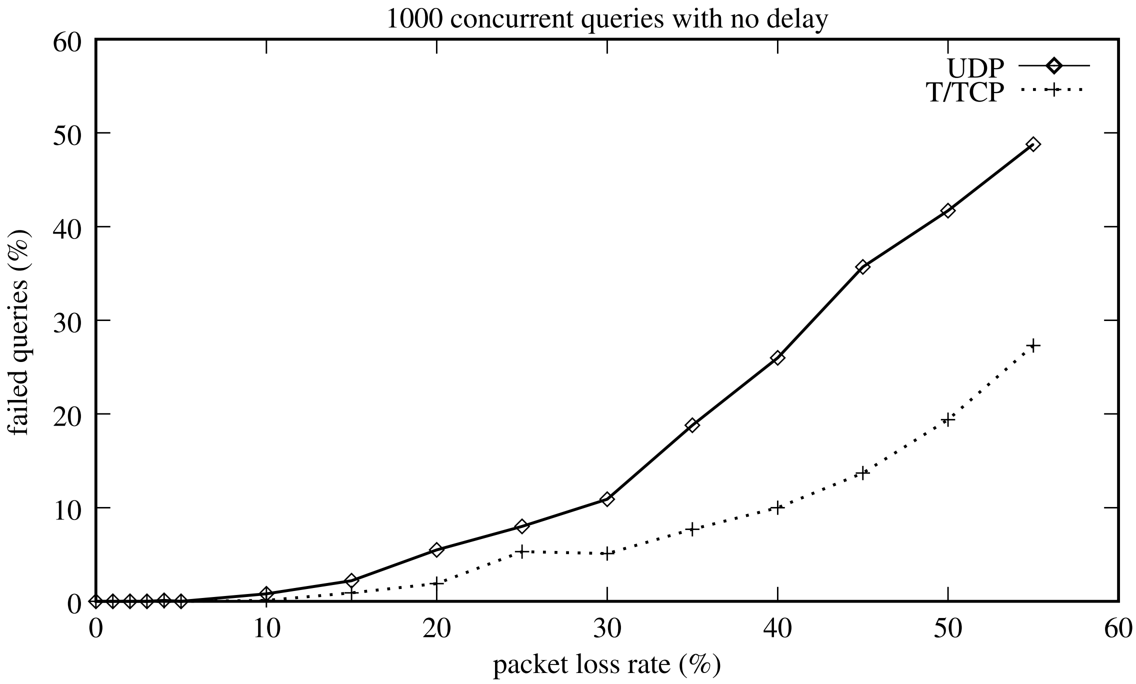

Figure 4.7 Query failure rates of UDP and T/TCP of 1000 concurrent queries for link with no delay

conducted to evaluate how the delay affects the query failure rates.

Figures 4.7 and 4.8 show the results. In either delay-time case, UDP and T/TCP showed little difference for how the rate of failed queries increased as the packet loss rate did. This suggests that the 500-millisecond delay time has little effect for the measured failure rate values. The author observed that on some packet loss rate values, the query failure rate values did not monotonically increased, such as those of T/TCP on the packet loss rate of 25%. The author considers this behavior as a result of probabilistic bias and divergence, since in the result of a preliminary test using 100 concurrent queries, the author observed much higher values of non-monotonic value changes.

UDP and T/TCP showed no failed queries when the packet loss rates ≤ 5%. As the packet loss rate increased, the difference between UDP and T/TCP results also increased, and the query failure-rate values of UDP were always larger than those of T/TCP. At the packet loss rate ≥ 30%, the values of query-failure rates for UDP is about twice as much as those of T/TCP.

The results indicate T/TCP is effective for decreasing the worst-case failure rate for DNS queries in the networks of high packet loss rates.

The author also evaluated how the difference of query completion time between the successful queries of T/TCP and UDP changes as the packet loss rate increases. 100 concurrent queries were conducted for each random packet loss rate value.

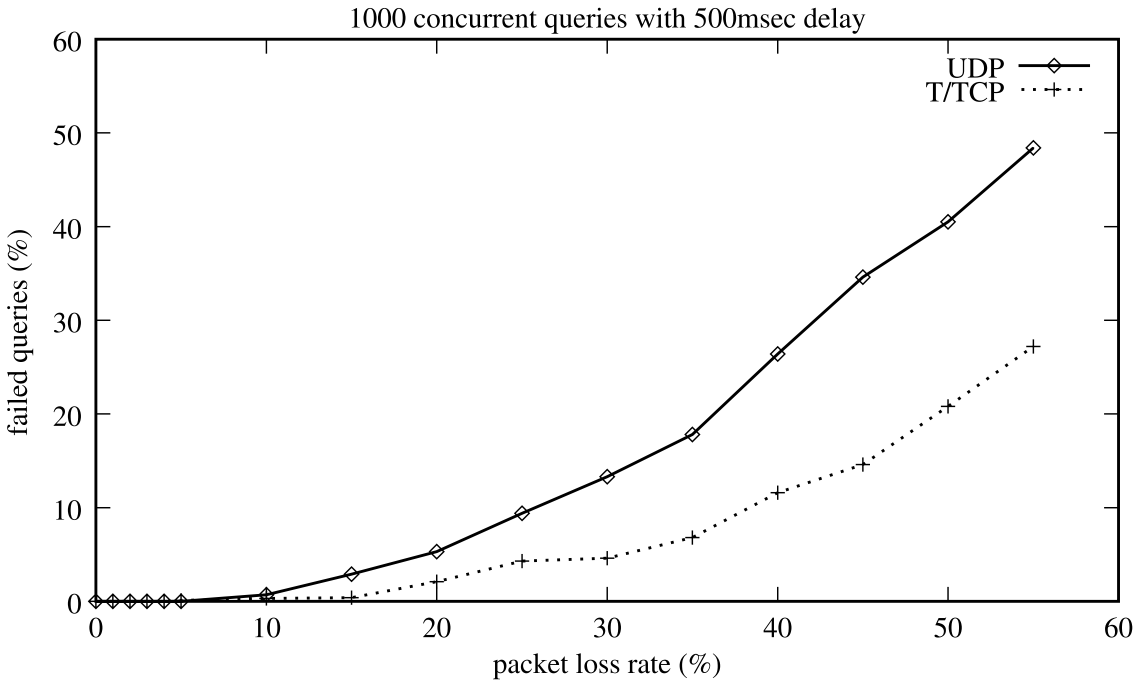

Figure 4.8 Query failure rates of UDP and T/TCP of 1000 concurrent queries for link with 500ms delay

Figure 4.9 shows the numbers of successful packets for UDP categorized by the retries. It indicates that more than 90% of the successful UDP queries completes within single retry attempts. Note that the completion time for many of *failed* queries are shorter than the successful queries, due to the retransmission algorithm described in Section 2.2.5.2.

Figure 4.10 shows the distributions of the completion time in T/TCP queries. It indicates that the minimum completion time increases as the packet loss rate does, but T/TCP still retains the TCP characteristics of exponential distribution of the packet retransmission. Note that on T/TCP the completion time of failed query is always longer than that of the successful queries, since TCP will retry until a given timeout is reached.

4.4 Concluding Remarks

In this chapter, the author proposed to use T/TCP as a DNS transport, evaluated the protocol by an implementation, and showed that T/TCP is an effective alternative to enhance the overall system security by increasing the reliability of the query processing especially for mobile equipments, and giving another choice of configuring firewalls.

The author lists some possible application fields of T/TCP to improve DNS Security. The key issues are to avoid UDP queries whenever possible, for minimizing the risk of UDP-related

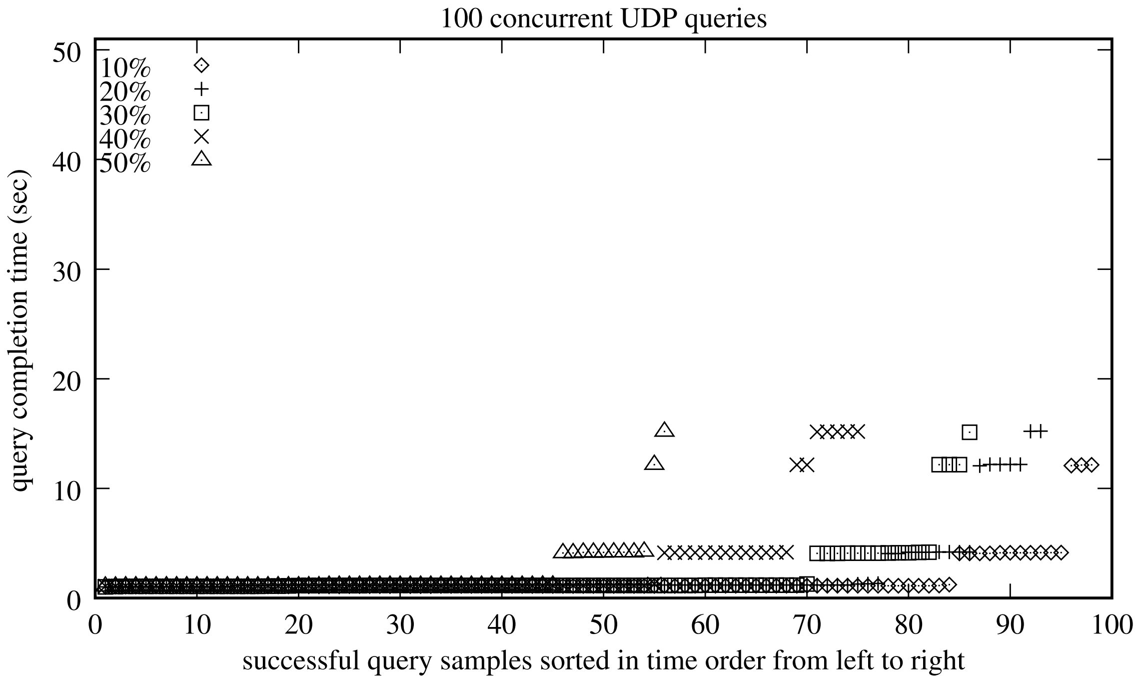

Figure 4.9 Completion time of successful UDP queries for 100 concurrent queries on 500ms delay link for different packet loss rates

attacks and to increase the rate of successful queries, while minimizing the overhead and the risk of attacks newly imposed by the T/TCP.

4.4.1 DNS for Mobile Equipments

DNS lookups from mobile equipments, such as from a notebook computer in a car using a cellular phone link, often fails because of the high packet loss rate. As shown in Section 4.3.4, T/TCP work better than UDP in such a case. Even in a lower packet loss rate, T/TCP has only 10% of query time overhead than UDP in a wide-area network environment which has the RTT of $\geq$ 60 milliseconds, as shown in Section 4.3.2 and described in Section 4.2.2. Changing the current UDP queries to T/TCP is a practical solution for mobile equipments, since it eliminates a requirement for UDP protocol stack and gives more control on the firewall between Internet and the networks of the equipments.

4.4.2 Inter-firewall DNS Exchange

DNS has become the only mandatorily-required UDP protocol which a firewall connected to the public Internet must support for non-private exchange. While simply prohibiting the UDP queries may work, it will increase the consumption of the server host resources, as TCP ex-

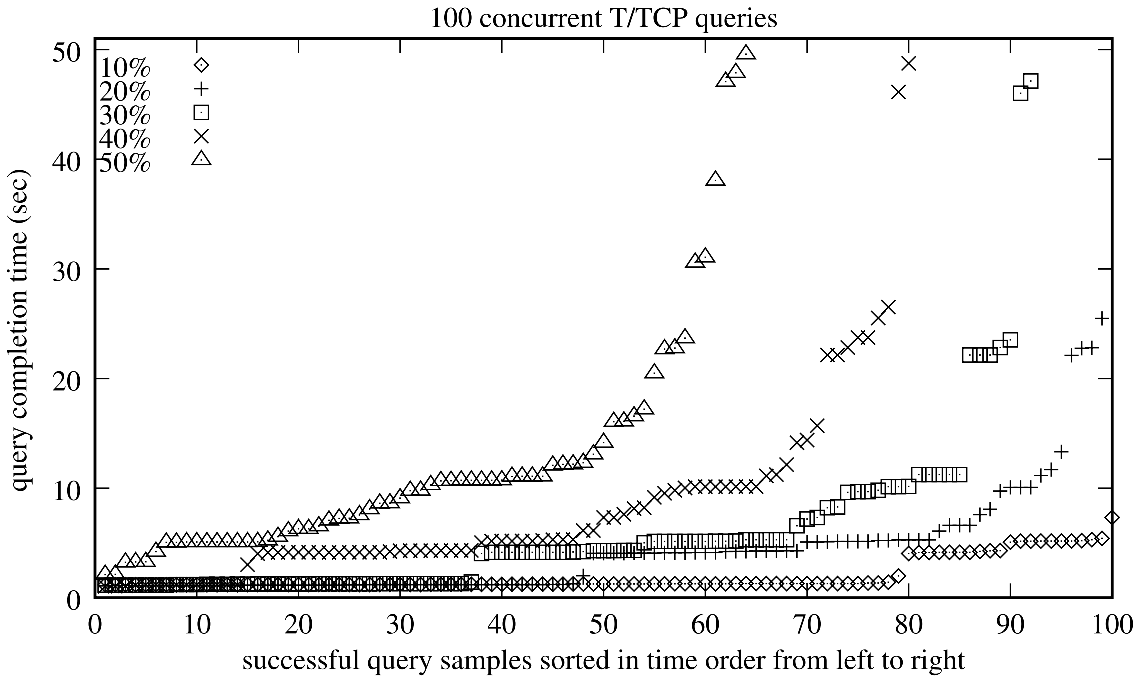

Figure 4.10 Completion time of successful T/TCP queries for 100 concurrent queries on 500ms delay link for different packet loss rates

change requires the connection blocks inside the operating system kernel. As shown in Section 4.3.3, T/TCP shortens the timeout state length, which largely affects the Maximon processing capability of a server host, to 2/15 of the traditional TCP. This will reduce the average resource consumption of the server host. And as shown in Section 4.3.2, T/TCP is a practical solution to replace UDP DNS lookups on an ADSL or other kinds of similar networks of larger latencies, which many of the end-user Internet sites use. The zone transfer exchange of DNS will benefit from T/TCP for the fast startup and earlier closing of connections as well. If the workload increase due to the T/TCP resource consumption is of concern, the practices for Web servers are applicable to reduce the impact.

4.4.3 Future Works

The author considers two major issues should be discussed for the further works: the detailed security analysis on T/TCP, and how T/TCP affects other applications, especially those based on UDP.

Chapter 5

Conclusion

5.1 Concluding Remarks

DNS is one of the core subsystems of the Internet. Every user of Internet takes DNS for granted since almost every object including URL (Uniform Resource Locator) [85], e-mail address, and host names are built into DNS using domain names.

DNS is, however, a set of distributed servers connected through Internet itself, and retains the surprisingly high availability despite of the pervasive mistakes and failures of configuration on each DNS server, resolver, and other components, which may affect the availavility of zone information [46, 86]. The author believes that the simplicity and robustness of DNS data transport is the key of this high availability, which leads into the reliability and the security as a goal.

In this dissertation, the author focused on the transport protocol issues, and specifically chose the UDP payload size limitation of DNS to find out how largely it might affect the migration or evolution from IPv4 to IPv6, and how the limitation could be worked around while minimizing the migration cost by using the existing protocol enhancements, such as EDNS0 and T/TCP, and showed that those enhancements are effective to solve the payload size problem, if widely and properly installed into the actual production systems.

The quality of DNS subsystems are heavily dependent on the personal skills of the administrator. The author, who has been a DNS administrator for more than 12 years since 1992, have had very hard time negotiating DNS-related issues with other administrators, since each of them has a different view from the others. A working document among IETF `dnsop` Working Group [50] also shows that it was very hard to reach for a rough consensus between DNS administrators.

Many operating documents such as RFCs and work-in-progress documents such as Internet-drafts are available for describing DNS as a set of reliable references, which is also publicly available and easily accessible through Internet. On the other hand, finding a journal

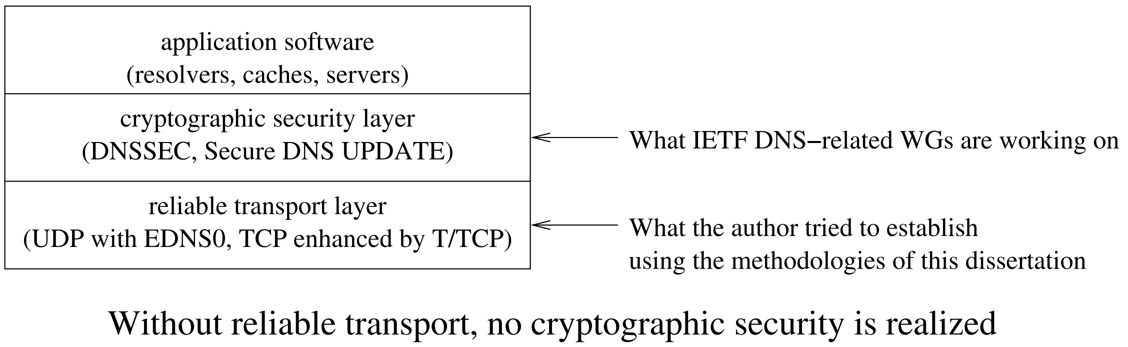

Figure 5.1 Reliable transport has a higher priority for development than a cryptographic security protocol

research paper related to the topics of this dissertation was not an easy task because of many reasons including that still a large number of DNS issues are remained to be solved, and that writing a document for a whole DNS requires a broad knowledge and skills of actual DNS operation and behavioral characteristics.

One of the reasons that DNS administration is dependent on the administrator is that description of DNS is not formally done only by RFCs but with a great help from practical expertise built into running source code and the skills of the administrators. Even if an apparent solution is found on DNS, making it as a part of production-level system may take many years, since the change of DNS core protocols should be very carefully planned and executed. A good example is the failure of historic SIG-based DNSSEC deployment, due to lack of the key management and distribution specification, took almost 7 years from January 1997 when the first proposal was published as RFC2065 (now superseded by RFC2535), to the DS-based DNSSEC was proposed in December 2003 as RFC3658. In this dissertation, the author set DNSSEC out of scope of the analysis and evaluation, since no cryptographic security protocol would work without a reliable transport foundation, as shown in Figure 5.1.

The author believes, nevertheless, that showing possible solutions, their analysis and evaluation, are what the researchers of Internet should do and practice, rather than proposing theoretical models which are theoretically complete but have very little chance to be actually chosen for a production-level system.

In this dissertation, the author intended to describe the ongoing issues and perform fact-based analysis using the real-world traffic data, and evaluating new protocols by implementing them with his own hands. The author believes that the current trends of DNS-related working groups and Japanese DNS administrators as of December 2004 are heavily biased to BIND-based mindset, which could lead into an unfair evaluation to other DNS implementations. The author rather tried to explain his views by referring to RFC and `djbdns`, a practical alternative to BIND while maintaining high compatibility with BIND and RFC specifications, which he has been using for the production systems since the first version was released.

In Chapter 1, the author described the importance of DNS as a core subsystem of Internet, by showing the historical role, newly-emerging protocol extensions, and how DNS interacts with other protocols. The author also discussed the security issues and the overall outline of this dissertation.

In Chapter 2, the author described the architecture of DNS and the transport protocol, by showing the protocol layers and the descriptions in Section 2.2, including the history of popular implementations and the behavior of resolvers with or without caches for looking up the servers. In Section 2.2.4, the details of DNS transport protocol was explained, focused on the following subjects: payload format, recursive and non-recursive queries, the usage of transport, the algorithm of choosing UDP or TCP as the lower-layer transport protocol in the TCP/IP protocol suite, the Root Server's limitation imposed by the 512-byte UDP payload length limitation, and the UDP retransmission protocol performed by the application program. In Section 2.3, the migration issues from IPv4 to IPv6 was described, of the split zone data spaces and the autoconfiguration support. In Section 2.4, details of DNSSEC authentication extension was described as well as how the newly-introduced DS-based DNSSEC works. In Section 2.5, dynamic DNS update mechanisms, including DNS UPDATE were discussed. The author claimed all of these newly-emerged extensions would contribute to increase the length of DNS payloads.

In Chapter 3, the author discussed the issues during the migration from IPv4 to IPv6 due to the DNS UDP payload length limitation, and quantitatively analyzed how the issues affect the DNS traffics by simulating the payload length of added or changed RRs during and after the migration from IPv4 to IPv6, by using the real-world DNS database traffic data from a large-scale campus network of Osaka University. The author then proposed possible solutions such as EDNS0, and comparatively evaluated the levels of improvement for each solution. The author concluded that the percentage of DNS answers exceeding the 512-byte UDP payload size limit, including the additional records, increases from 0.04% to 1~3% during and after the migration from IPv4 to IPv6 with a simulation by packet-length recalculation. The author also presented the effectiveness of EDNS0 to prevent performance degradation by protocol fallback to TCP, and setting the maximum payload length to 4096 bytes by EDNS0 effectively prevents the DNS protocol fallback to TCP after the migration to IPv6 completes. The author also discussed the effectiveness of T/TCP, to prevent performance degradation by increase of number of packets exchanged during the resolver-server exchanges over TCP, and the usefulness of the T/TCP for

DNSSEC-based resolver-server exchanges over IPv6. The author also claimed that the limiting the payload length by selectively choosing the RRs to answer for a DNS query should not be considered as a solution and rather treated be a practical workaround to cope with the current 512-byte limitation of DNS UDP payload length.

In Chapter 4, the author described the T/TCP fundamentals as a TCP extension and the advantages to traditional TCP, including the DoS immunity added by T/TCP comparing it to UDP. The evaluation results of T/TCP used as a DNS transport was shown in Section 4.3, using the criteria of the protocol overhead and allocated connection blocks of the host, and measured that the response time for a query was reduced from 100% (twice as much) in traditional TCP to 10% more in T/TCP of that of UDP in a higher-latency link, and that `TIME_WAIT` length was reduced from from 60 to 8 seconds for FreeBSD 4.7-RELEASE. The author concluded that T/TCP was effective for providing DNS service for mobile equipments, inter-firewall DNS exchange, and other kinds of generic replacement of UDP DNS lookups.

The author hopes that the results of analysis, evaluation and experimentation given in this dissertation provides examples of the design and operation of the future DNS protocols.

5.2 Future Works

The long-term goal of this dissertation is to develop DNS with sufficient security to protect it against forgery or alteration of the RRs, as well as making it reliable enough to support future expansion of the servers and resolvers. The author also believes that gathering security-related information from DNS exchange is a practical and effective tool for monitoring the Internet activities and preventing furure security incidents. The author believes that the following issues should be investigated to make DNS secure and make it also a tool for making Internet more secure.

(1) Validation of effectiveness of EDNS0 and T/TCP on wide-area networks

For proving the effectiveness of EDNS0 and T/TCP on DNS proposed in this dissertation, validation through experiments on a large-scale wide-are network is essential. Performance measurement of actual running servers at large-scale ISPs (Internet Service Providers) and DNS server operating sites is needed to prove whether the proposed protocols are really effective or not. Computer simulation without generating actual traffic is also useful to fine-tune the protocol parameters before testing it on the production-system network. Protocols other than EDNS0 and T/TCP, such as SCTP, should also be evaluated in the same manners as well. DoS immunity analysis using the actual or simulated attacks should also be performed.

(2) Estimation of large-scale DNSSEC deployment and the DNS payload length increase

Since more and more hosts are dependent on DNS, such as Internet telephony or VoIP (Voice-over-IP) systems, the authentication of domain names is crucial for successfully deploying the service to the public. The VoIP systems are dependent on DNS to map IP addresses to the phone numbers, such as ENUM [87]. To provide a reliable service which no attacker can alter the mapping between the phone numbers and IP addresses, combining ENUM with DNSSEC is the best current practical solution. This means the average length of DNS payloads will be sharply increased as the VoIP systems become more popular. Analysis of the payload length and the effects to the DNS servers, such as that performed for a case of migration from IPv4 to IPv6, should be thoroughly done for a system design of VoIP services.

(3) Establishment of secure DNS monitoring method and the analyzing algorithms

DNS servers, especially of those running in ISPs, are practical points of monitoring Internet activity trends. The author conducted a behavioral analysis of DNS and TCP connections [88], and have found out that DNS query answers of the A RRs are highly likely to be followed by the TCP connection attempts to a given address among the A RRs. An anti-worm filtering approach [89] is proposed to block worms which does not depend on DNS server lookups, since most of the legitimate TCP traffics are using DNS. Discovering this sort of relationship between DNS traffic and other protocol traffics is useful for monitoring Internet and preventing the security incidents. Methodology of efficiently collecting DNS exchange data and analyzing the traffic securely without disclosing the privacy-related information such as a flow-graph-based analysis [90] should be established, especially for an effective multi-point monitoring of DNS server response times and the impact of server performance factors [91].

Acknowledgements

I would like to thank Professor Shinji Shimojo, of Cybermedia Center at Osaka University, for his overall and primary supervision, countless suggestions, and constructive comments on my research activities and writing this dissertation.

I would like to express my gratitude to Professor Shojiro Nishio, Professor Toru Fujiwara, Professor Norihisa Komoda, and Professor Fumio Kishino, of the Department of Multimedia Engineering in the Graduate School of Information Science and Technology at Osaka University, Associate Professor Ken-ichi Baba of Cybermedia Center at Osaka University, and Associate Professor Motonori Nakamura of Academic Center for Computing and Media Studies at Kyoto University, for their support on numerous suggestions for revising this dissertation.

My sincere appreciation goes to Dr. Hiroki Nogawa, Deputy Director of Information Center for Medical Sciences at Tokyo Medical and Dental University, for his active partnership role in my research during his career in Cybermedia Center at Osaka University.

I would also like to thank the members and alumni of the Shimojo Laboratory members of Cybermedia Center at Osaka University. Among the alumni, Mr. Toshiharu Otsuka and Mr. Kenji Nakano helped me a lot by performing prototype experiments of the research. My appreciation also goes to Assistant Professor Susumu Date of Department of Bioinformatics Engineering in the Graduate School of Information Science and Technology at Osaka University, and Dr. Yasushi Takagi of NTT Network Service Systems Laboatories, for providing their expertise and suggestion of writing doctoral dissertations.

My acknowledgement also goes to the operation staff members of ODINS (Osaka Daigaku Information Network System) for their kind assistance to build a traffic monitoring system in their network facility.

A major portion of this research was funded and supported by the KDDI R&D Laboratories, with the help of the Security Laboratory members. I would like to thank Mr. Tohru Asami, the president and CEO, for his continuous support and giving me a chance to study again as a doctoral candidate.

I would also like to thank Mr. Koji Nakao, the director of Information Security Department of KDDI Corporation and the group leader of Security Advancement Group in National

Institute of Information and Communication Technology (NICT), for his suggestions and managemental support.

My appreciation also go to the current and former Security Laboratory members of KDDI R&D Laboratories, including Mr. Toshiaki Tanaka, Dr. Fumiaki Sugaya, Mr. Yutaka Miyake, Mr. Yasuyuki Watanabe, and Dr. Keisuke Takemori.

I would like to show my sincere respect to Mr. Paul Vixie, an author of BIND DNS Server and the CEO of Internet Software Consortium, for his kind support and constructive criticisms to my research and operational activities.

My deep respect also goes to Dr. Akira Kato, Associate Professor at the University of Tokyo, for his constructive criticisms based on expertise and knowledge on DNS, Internet mail, and other operation of Internet services, during revision of this dissertation.

My sincere respect also goes to Dr. Daniel J. Bernstein, Associate Processor of Department of Computer Science in University of Illinois at Chicago, and also the author of `djbdns`, for enlighting me by providing an alternative reliable working set of DNS tools and software.

I would like to thank Dr. Eiiti Wada, Professor Emeritus at the University of Tokyo and the Research Director of IIJ Research Laboratory, for his leadership and patience while I was studying as a member of Information Processing Laboratory in Graduate School of Information Engineering at the University of Tokyo. My appreciation also goes to the laboratory alumni.

I would also like to thank Dr. Shin-ichi Nakagawa and Dr. Hiroyuki Ohno of NICT, and Dr. Mieko Kimura of Takeda Research Institute of Life Science and Medicine, for realizing me the importance of obtaining the degree of Ph.D.

My appreciation also goes to Mr. Kazuo Hirono of Profile Co. Ltd., who has been a long-time mentor of my research activities, and to Mr. Yutaka Sakurai of Cisco Systems K. K., for his continuous encouragement and support during my research activities.

This dissertation is typeset with the LaTeX style file designed by Dr. Haruhiko Okumura of Course for Information Science Education in Faculty of Education at Mie University. His LaTeX expertise largely contributed to improve the typeset quality of this dissertation.

Finally I must show my deepest gratitude to my family members, first to my father Tsuneji, who suddenly passed away on August 22nd, 2004, who taught me the fundamental discipline and principles of doing scientific research as well as the other essential rules of life; also to my mother Kiyoko, who raised me with her courteous parentalship and emotional support; and to my wife Kyoko, Associate Professor of Faculty of Language and Culture at Osaka University and a German language specialist, who always encourages me to keep on doing my job with her happiest smile, wise suggestions, and continuous assistance every day.

References

[1] Mockapetris, P. V.: Domain names – concepts and facilities (1983). RFC883 (superceded by RFC1034).

[2] Mockapetris, P. V.: Domain names – concepts and facilities (1987). RFC1034 (also STD13).

[3] Mockapetris, P. V.: Domain names – implementation and specification (1987). RFC1035 (also STD13).

[4] R. Braden (Editor): Requirements for Internet Hosts – Application and Support (1989). RFC1123.

[5] Elz, R. and Bush, R.: Clarification to the DNS Specification (1997). RFC2181.

[6] Vixie, P.: Extension Mechanisms for DNS (EDNS0) (1999). RFC2671.

[7] Thomson, S., Huitema, C., Ksinant, V. and Souissi, M.: DNS Extensions to support IP version 6 (2003). RFC3596.

[8] Eastlake, D.: Domain Name System Security Extensions (1999). RFC2535.

[9] Vixie, P., Thomson, S., Rekhter, Y. and Bound, J.: Dynamic Updates in the Domain Name System (DNS UPDATE) (1997). RFC2136.

[10] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1 (1999). RFC2616.

[11] Dierks, T. and Karlton, P. L.: The TLS Protocol Version 1.0 (1999). RFC2246.

[12] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J. and Wright, T.: Transport Layer Security (TLS) Extenstions (2003). RFC3546.

[13] Postel, J.: Transmission Control Protocol (1981). RFC793 (also STD7).

[14] Kent, S. and Atkinson, R.: Security Architecture for the Internet Protocol (1998). RFC2401.

[15] Postel, J.: User Datagram Protocol (1980). RFC768 (also STD6).

[16] Gilmore, J.: AS IF: draft-ietf-dnsext-ad-is-secure-03.txt (2001). The Risks Digest Vol. 21: Issue 56, `http://catless.ncl.ac.uk/Risks/21.56.html#subj7`.

[17] Vixie, P.: a message of IETF `dnsext` Working Group Mailing List (2002). `Message-Id: <20021121053500.8C17737A037@as.vix.com>`, accessible from IETF Web site

http://www.ietf.org/.

[18] Internet Software Consortium: BIND. http://www.isc.org/bind/.

[19] Bernstein, D. J.: djbdns. http://cr.yp.to/djbdns.html.

[20] NLnet Labs: Name Server Daemon (NSD). http://www.nlnetlabs.nl/nsd/.

[21] Braden, R.: T/TCP – TCP Extensions for Transactions Functional Specification (1994). RFC1644.

[22] Gummadi, K. P., Saroiu, S. and Gribble, S. D.: King: Estimating Latency between Arbitrary Internet End Hosts, *Proceedings of the Second ACM SIGCOMM Internet Measurement Workshop (IMW2002)*, pp. 5–18 (2002). ISBN 1-58113-603-X/02/0011.

[23] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and Paxson, V.: Stream Control Transmission Protocol (2000). RFC2960.

[24] Stone, J., Stewart, R. and Otis, D.: Stream Control Transmission Protocol (SCTP) Checksum Change (2002). RFC3309.

[25] Rivest, R.: The MD5 Message-Digest Algorithm (1992). RFC1321.

[26] Rekhter, Y. and Li, T.: A Border Gateway Protocol 4 (BGP-4) (1995). RFC1771.

[27] Bellovin, S.: Defending Against Sequence Number Attacks (1996). RFC1948.

[28] Wellington, B.: Secure Domain Name System (DNS) Dynamic Update (2000). RFC3007.

[29] The FreeBSD Project: FreeBSD. http://www.freebsd.org/.

[30] The OpenBSD Project: OpenBSD. http://www.openbsd.org/.

[31] The NetBSD Project: NetBSD. http://www.netbsd.org/.

[32] Linux Headquarters Inc.: LinuxHQ. http://www.linuxhq.com/.

[33] CERT/CC: Buffer Overflows in Multiple DNS Resolver Libraries. CERT Advisory CA-2002-19, http://www.cert.org/advisories/CA-2002-19.html.

[34] CERT/CC: Multiple vendors' Domain Name System (DNS) stub resolvers vulnerable to buffer overflows. CERT Vulnerability Note VU#803539, http://www.kb.cert.org/vuls/id/803539.

[35] CERT/CC: Domain Name System (DNS) stub resolver libraries vulnerable to buffer overflows via network name or address lookups. CERT Vulnerability Note VU#844360, http://www.kb.cert.org/vuls/id/844360.

[36] The FreeBSD Project: Buffer Overflow in Resolver. FreeBSD Security Advisory FreeBSD-SA-02:28.resolv.

[37] CERT/CC: Multiple Vulnerabilities in BIND. CERT Advisory CA-2002-31, http://www.cert.org/advisories/CA-2002-31.html.

[38] CERT/CC: ISC BIND 8 fails to properly dereference cache SIG RR elements with invalid

expiry times from the internal database. CERT Vulnerability Note VU#581682, `http://www.kb.cert.org/vuls/id/581682`.

[39] CERT/CC: Cached malformed SIG record buffer overflow. CERT Vulnerability Note VU#852283, `http://www.kb.cert.org/vuls/id/852283`.

[40] CERT/CC: Denial-of-Service Vulnerability in ISC BIND 9. CERT Advisory CA-2002-15.

[41] Vixie, P., Gudmundsson, O., Eastlake, D. and Wellington, B.: Secure Key Transaction Authentication for DNS (TSIG) (2000). RFC2845.

[42] Vixie, P. and Kato, A.: Modern DNS as a Coherent Dynamic Universal Database, *IEICE Trans. Commun. (Japanese Edition)*, Vol. J87-B, No. 10, pp. 1534–1541 (2004).

[43] Vixie, P. and Mail Abuse Prevention System, LLC: MAPS Realtime Blackhole List (RBL) Overview. `http://www.mail-abuse.com/services/mds_rbl.html`.

[44] Ohta, M.: Incremental Zone Transfer in DNS (1996). RFC1995.

[45] Vixie, P.: A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY) (1996). RFC1996.

[46] Jung, J., Sit, E., Balakrishnan, H. and Morris, R.: DNS Performance and the Effectiveness of Caching, *Proceedings of the First ACM SIGCOMM Internet Measurement Workshop (IMW2001)*, pp. 153–167 (2001). ISBN 1-58113-435-5/01/0011.

[47] National Infrastructure Security Co-ordination Centre: Vulnerability Issues in Implementations of the DNS Protocol. NISCC Vulnerability Advisory 758884/NISCC/DNS, `http://www.cert.org/advisories/CA-2002-19.html`.

[48] Gudmundsson, O.: DNSSEC and IPv6 A6 aware server/resolver message size requirements (2001). RFC3226.

[49] Bernstein, D. J.: User's guide to name resolution. `http://cr.yp.to/djbdns/resolve.html`.

[50] Durand, A., Ihren, J. and Savola, P.: Operational Considerations and Issues with IPv6 DNS (2004). INTERNET-DRAFT draft-ietf-dnsop-ipv6-dns-issues-08.txt.

[51] Durand, A. and Ihren, J.: DNS IPv6 transport operational guidelines (2004). INTERNET-DRAFT draft-ietf-dnsop-ipv6-transport-guidelines-02.txt.

[52] R. Droms (Editor), Bound, J., Volz, B., Lemon, T., Perkins, C. and Carney, M.: Dynamic Host Configuration Protocol for IPv6 (DHCPv6) (2003). RFC3315.

[53] Droms, R.: Dynamic Host Configuration Protocol (1997). RFC2131.

[54] Thomson, S. and Narten, T.: IPv6 Stateless Address Autoconfiguration (1998). RFC2462.

[55] Eastlake, D.: DNS Request and Transaction Signatures (SIG(0)s) (2000). RFC2931.

[56] Baba, T., Kusaka, T., Yamaoka, M. and Matsuda, S.: Implementation and Estimation of a Domain Name System with Access Control, *IPSJ SIG Technical Report 2004-CSEC-24*,

Vol. 2004, No. 22, pp. 99–104 (2004).

[57] Gudmundsson, O.: Delegation Signer (DS) Resource Record (RR) (2003). RFC3658.

[58] Japan Registry Service (JPRS): DNSSEC Cho-Nyumon (DNSSEC for the Dummies) Version 1.1 (2004). http://etjp.jp/about/wg/dnssec/dnssec-cho-nyumon.pdf (written in Japanese).

[59] Dynamic Network Services, Inc.: DynDNS.org – Developers – Update Specifications. http://www.dyndns.org/developers/specs/.

[60] Rikitake, K., Kikuchi, T., Nagata, H., Hamai, T. and Asami, T.: Practical DNS Support for Dialup ADSL, *Proceedings of IPSJ Computer Security Symposium 2001 (CSS2001)*, IPSJ Symposium Series, Vol. 2001, No. 15, IPSJ, pp. 73–79 (2001).

[61] Rikitake, K., Kikuchi, T., Nagata, H., Hamai, T. and Asami, T.: Security Issues on Home Teleworking over Internet, *IEICE Technical Report IA2001-20*, Vol. 101, No. 440, pp. 9–16 (2001).

[62] The OpenBSD Project: OpenSSH. http://www.openssh.org/.

[63] Rikitake, K., Nogawa, H., Tanaka, T., Nakao, K. and Shimojo, S.: DNS Transport Size Issues in IPv6 Environment, *Proceedings of the 2004 International Symposium of Applications and the Internet (SAINT2004) Workshops*, pp. 141–145 (2004). ISBN 0-7695-2050-2/04.

[64] Rikitake, K., Nogawa, H., Tanaka, T., Nakao, K. and Shimojo, S.: An Analysis of DNS Payload Length Increase during Transition to IPv6, *IEICE Trans. Commun. (Japanese Edition)*, Vol. J87-B, No. 10, pp. 1552–1563 (2004).

[65] Thomson, S. and Huitema, C.: DNS Extensions to support IP version 6 (1995). RFC1886.

[66] Crawford, M. and Huitema, C.: DNS Extensions to Support IPv6 Address Aggregation and Renumbering (2000). RFC2874.

[67] Bush, R., Durand, A., Fink, B., Gudmundsson, O. and Hain, T.: Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS) (2002). RFC3363.

[68] Austein, R.: Tradeoffs in Domain Name System (DNS) Support for Internet Protocol version 6 (IPv6) (2002). RFC3364.

[69] M. Roesch et al.: Snort. http://www.snort.org/.

[70] TCPDUMP Public Repository: tcpdump. http://www.tcpdump.org/.

[71] Eastlake, D.: Secret Key Establishment for DNS (TKEY RR) (2000). RFC2930.

[72] Japan Registry Service (JPRS): On Maximum Number of DNS Servers (2003). http://jprs.jp/tech/jp-dns-info/2003-07-10-max-number-of-dns-server.html (written in Japanese).

[73] Microsoft Corporation: Using Extension Mechanisms for DNS (EDNS0).

http://www.microsoft.com/resources/documentation/WindowsServ/2003/standard/proddocs/en-us/sag_DNS_imp_EDNSsupport.asp.

[74] Deering, S. and Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification (1998). RFC2460.

[75] Rikitake, K., Nakao, K., Nogawa, H. and Shimojo, S.: T/TCP for DNS: A Performance and Security Analysis, *IPSJ Journal*, Vol. 44, No. 8, pp. 2060–2071 (2003).

[76] L. Ren and J. Zhang: T/TCP for Linux. http://ttcplinux.sourceforge.net/.

[77] Vixie, P. and Kato, A.: DNS Response Size Issues (2004). INTERNET-DRAFT draft-ietf-dnsop-respsize-01.txt.

[78] Kato, A. and Sekiya, Y.: Analysis of DNS Traffic at a DNS Server in an ISP, *IEICE Trans. Commun. (Japanese Edition)*, Vol. J87-B, No. 3, pp. 327–335 (2004).

[79] Sekiya, Y., Cho, K., Kato, A. and Murai, J.: Research of Method for DNS Performance Measurement and Evaluation Based on Benchmark DNS Servers, *IEICE Trans. Commun. (Japanese Edition)*, Vol. J87-B, No. 10, pp. 1542–1551 (2004).

[80] Braden, R.: Extending TCP for Transactions – Concepts (1992). RFC1379.

[81] Stevens, W. R.: Part 1: TCP for Transactions, *TCP/IP Illustrated, Volume 3*, Addison–Wesley, pp. 3–158 (1996).

[82] Lemon, J.: Resisting SYN flood DoS attacks with a SYN cache. http://people.freebsd.org/˜jlemon/papers/syncache.pdf.

[83] The FreeBSD Project: The ttcp(4) man page. Available as a part of FreeBSD distributions.

[84] Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols, *Computer Communication Review*, Vol. 27, No. 1, pp. 31–41 (1997).

[85] Berners-Lee, T., Masinter, L. and McCahill, M.: Uniform Resource Locators (URL) (1994). RFC1738.

[86] Pappas, V., Xu, Z., Lu, S., Massey, D., Terzis, A. and Zhang, L.: Impact of Configuration Errors on DNS Robustness, *Computer Communication Review*, Vol. 34, No. 4, pp. 319–330 (2004). Proceedings of SIGCOMM 2004.

[87] Faltstrom, P. and Mealling, M.: The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDDS) Application (ENUM) (2004). RFC3761.

[88] Rikitake, K., Nogawa, H., Tanaka, T., Nakao, K. and Shimojo, S.: Behavioral Analysis of DNS and TCP Connections, *Proceedings of IPSJ Computer Security Symposium 2003 (CSS2003)*, IPSJ Symposium Series, Vol. 2003, No. 15, IPSJ, pp. 521–526 (2003).

[89] Okamoto, T.: Packet Filtering Using DNS Responses against Worm Propagation, *IPSJ Journal*, Vol. 45, No. 10, pp. 2407–2415 (2004).

[90] Cranor, C. D., Gansner, E., Krishnamurthy, B. and Spatscheck, O.: Characterizing large DNS traces using graphs, *Proceedings of the First ACM SIGCOMM Internet Measurement Workshop (IMW2001)*, pp. 55–67 (2001). ISBN 1-58113-435-5/01/0011.

[91] Liston, R., Srinivasan, S. and Zegura, E.: Diversity in DNS Performance Measures, *Proceedings of the Second ACM SIGCOMM Internet Measurement Workshop (IMW2002)*, pp. 19–31 (2002). ISBN 1-58113-603-X/02/0011.

List of Publications by the Author

A. Journal Papers

1. Rikitake, K., Nogawa, H., Tanaka, T., Nakao, K. and Shimojo, S.: An Analysis of DNS Payload Length Increase during Transition to IPv6, *IEICE Trans. Commun. (Japanese Edition)*, Vol. J87-B, No. 10, pp. 1552–1563 (2004).
2. Rikitake, K., Nakao, K., Nogawa, H. and Shimojo, S.: T/TCP for DNS: A Performance and Security Analysis, *IPSJ Journal*, Vol. 44, No. 8, pp. 2060–2071 (2003).
3. Takemori, K., Rikitake, K., Miyake, Y. and Nakao, K.: Implementation of Dynamic Diversion Mechanisms for Secure and Effective Logging on Intrusion Trap System, *IPSJ Journal*, Vol. 44, No. 8, pp. 1838–1847 (2003).

B. International Conference Papers

1. Rikitake, K., Nogawa, H., Tanaka, T., Nakao, K. and Shimojo, S.: DNS Transport Size Issues in IPv6 Environment, *Proceedings of the 2004 International Symposium of Applications and the Internet (SAINT2004) Workshops*, pp. 141–145 (2004). ISBN 0-7695-2050-2/04.
2. Takemori, K., Rikitake, K., Miyake, Y. and Nakao, K.: Intrusion Trap System: An Efficient Platform for Gathering Intrusion-related Information, *Proceedings of the 10th International Conference on Telecommunications (ICT2003)*, IEEE, pp. 614–619 (2003). ISBN 0-7803-7661-7.
3. Rikitake, K.: Breaking Barriers to Popularize Internet Streaming Broadcast, *Proceedings CD-ROM of INET2000 The Internet Global Summit*, Internet Society (2000). ISBN 1-891562-09-6, `http://www.isoc.org/inet2000/cdproceedings/8g/8g_1.htm`.
4. Rikitake, K.: MENVPRIV: A User-Controllable Privacy-Enhanced E-Mail Network, *Program Book of Abstracts, International Conference on Privacy*, pp. 152–153 (1997). Montreal, Québec, Canada.
5. Rikitake, K.: MAILFEED: A POP3-based Inter-domain Mail-Forwarding System, *The Proceedings of Internet Conference 1996*, JSSST, pp. 3–10 (1996). JSSST Symposium

Proceeding Series No.3, ISSN 1341-870X.

C. Domestic Conference Papers

1. Rikitake, K., Nogawa, H., Tanaka, T., Nakao, K. and Shimojo, S.: Internet Security Management on Teleworking Environment, *Proceedings of the Sixth Japan Telework Society Conference*, Japan Telework Society, pp. 85–90 (2004).
2. Rikitake, K.: DNS Reliability and the Security Issues, *The Summary Symposium of Developing Human Resource for Building Secure Networks*, Cybermedia Center, Osaka University, pp. 101–115 (2003).
3. Rikitake, K., Nogawa, H., Tanaka, T., Nakao, K. and Shimojo, S.: Behavioral Analysis of DNS and TCP Connections, *Proceedings of IPSJ Computer Security Symposium 2003 (CSS2003)*, IPSJ Symposium Series, Vol. 2003, No. 15, IPSJ, pp. 521–526 (2003).
4. Rikitake, K., Sugaya, F., Nakao, K., Nogawa, H. and Shimojo, S.: Resource Consumption Analysis of DNS Servers against DoS Attacks, *IPSJ SIG Technical Reports 2003-QAI-8*, Vol. 2003, No. 68, pp. 51–54 (2003).
5. Rikitake, K., Nakao, K., Nogawa, H. and Shimojo, S.: Securing Public DNS Communication, *IPSJ SIG Notes 2003-CSEC-20*, Vol. 2003, pp. 179–184 (2003).
6. Rikitake, K., Takemori, K., Miyake, Y., Nakao, K., Nogawa, H. and Shimojo, S.: Design of Robust DNS by Intrusion Detection, *Proceedings of IPSJ Computer Security Symposium 2002 (CSS2002)*, IPSJ Symposium Series, Vol. 2002, No. 16, IPSJ, pp. 17–22 (2002).
7. Rikitake, K., Kikuchi, T., Nagata, H., Hamai, T. and Asami, T.: Secure Gateway System Design for Home Teleworking, *IPSJ SIG Notes 2002-CSEC-17*, Vol. 2002, pp. 1–6 (2002).
8. Rikitake, K., Kikuchi, T., Nagata, H., Hamai, T. and Asami, T.: Secure Teleworking over Wireless Internet, *IEICE General Conference Symposium SB-12-3*, IEICE (2002).
9. Rikitake, K., Kikuchi, T., Nagata, H., Hamai, T. and Asami, T.: Defending Servers by Randomizing Listening Port Numbers, *IPSJ SIG Notes 2001-CSEC-15*, Vol. 2001, pp. 7–12 (2001).
10. Kikuchi, T., Asami, T., Rikitake, K., Nagata, H. and Hamai, T.: Reverse-Path-Based Filtering against IP Address Spoofing, *Proceedings of IPSJ Computer Security Symposium 2001 (CSS2001)*, IPSJ Symposium Series, Vol. 2001, No. 15, IPSJ, pp. 191–196 (2001).
11. Takemori, K., Rikitake, K., Kiyomoto, S., Tanaka, T. and Nakao, K.: Implementation and Evaluation of Intrusion Trap System, *Proceedings of IPSJ Computer Security Sym-*

posium 2001 (CSS2001), IPSJ Symposium Series, Vol. 2001, No. 15, IPSJ, pp. 415–420 (2001).

12. Rikitake, K., Kikuchi, T., Nagata, H., Hamai, T. and Asami, T.: Practical DNS Support for Dialup ADSL, *Proceedings of IPSJ Computer Security Symposium 2001 (CSS2001)*, IPSJ Symposium Series, Vol. 2001, No. 15, IPSJ, pp. 73–79 (2001).
13. Rikitake, K., Kikuchi, T., Nagata, H., Hamai, T. and Asami, T.: Security Issues on Home Teleworking over Internet, *IEICE Technical Report IA2001-20*, Vol. 101, No. 440, pp. 9–16 (2001).
14. Rikitake, K., Kikuchi, T., Nagata, H., Hashimoto, K. and Asami, T.: Solving Management Issues of Inexpensive Internet-VPN Teleworking (written in Japanese), *Human Interface Society HIS2001 Symposium Proceedings*, Human Interface Society, pp. 593–596 (2001).
15. Takemori, K., Rikitake, K., Kiyomoto, S., Tanaka, T. and Nakao, K.: Design and Implementation of Intrusion Trap System, IPSJ Symposium Series, Vol. 3, IPSJ, pp. 495–496 (2001). Paper 2G-1.
16. Rikitake, K., Kikuchi, T., Nagata, H. and Asami, T.: Information Security Management under Teleworking Environment (written in Japanese), *IPSJ 63rd National Convention Proceedings*, Vol. 3, IPSJ, pp. 625–628 (2001). Paper 2B-2 (awarded as a Best Paper of the Convention).

D. Books and Magazine Articles

(All books and articles here are written in Japanese)

1. Rikitake, K.: Understanding Security Technology for Enterprise Internet (2001). *Nikkei Computer* Magazine, Nikkei Business Publications, Inc., October 8, 2001 – March 25, 2002 issues, 13 articles in series.
2. Rikitake, K.: Risk Assessment for the Internet Era (2001). *Nikkei Computer* Magazine, Nikkei Business Publications, Inc., April 9, 2001 – September 24, 2001 issues, 13 articles in series.
3. Rikitake, K.: Reconsidering TCP/IP as A Solution (2000). *ASCII Network* Magazine, ASCII Publishing, March 2000 – December 2000 issues, 10 articles in series.
4. Rikitake, K.: *Carelessly-implemented Copy Protection Hampers Popularization of Internet Streaming* (2000). H. Ohsawa (ed.), *Internet Streaming*, Kyoritsu Shuppan, p. 221, ISBN 4-320-02978-X.
5. Rikitake, K.: Understanding Latest Internet Technologies (2000). *Nikkei Computer*

Magazine, Nikkei Business Publications, Inc., January 3, 2000 – September 25, 2000 issues, 20 articles in series.

6. Rikitake, K.: TCP/IP Network Management (1999). *ASCII NT* Magazine, ASCII Publishing, August 1999 – November 1999 issues, 4 articles in series.
7. Rikitake, K.: Learning Internet Troubleshooting (1999). *Nikkei Computer* Magazine, Nikkei Business Publications, Inc., April 12, 1999 – September 27, 1999 issues, 13 articles in series.
8. Rikitake, K.: Learning Practical Internet Technologies (1998). *Nikkei Computer* Magazine, Nikkei Business Publications, Inc., October 12, 1998 – March 29, 1999 issues, 13 articles in series.
9. Rikitake, K.: Basic TCP/IP Technology (1999). *ASCII NT* Magazine, ASCII Publishing, January 1999 – June 1999 issues, 6 articles in series.
10. Rikitake, K.: The "Standardization" of Internet (1999). *Dr. Dobb's Journal Japan*, Shoeisha, January 1999, pp. 20–26 .
11. Rikitake, K.: Learning Basic Internet Technologies (1998). *Nikkei Computer* Magazine, Nikkei Business Publications, Inc., April 11, 1998 – September 28, 1998 issues, 13 articles in series.
12. Rikitake, K.: qmail Basics and Configuration (1998). *Dr. Dobb's Journal Japan*, Shoeisha, January 1998, pp. 134–146.
13. Rikitake, K.: *Professional Internet*, Ohmsha (1998).
14. Rikitake, K.: Internet Kaleidoscope (1997). *Internet@ASCII* Magazine, ASCII Publishing, May 1997 – August 1998 issues, 16 articles in series.
15. Rikitake, K.: Understanding Internet in 3 Minutes (1996). *Internet@ASCII* Magazine, ASCII Publishing, May 1996 – April 1997 issues, 12 articles in series.
16. Schneier, B., K. Rikitake (Japanese Translation Supervisor) and N. Michishita (Japanese Translator): *E-Mail Security (Japanese Version)*, Ohmsha (1995).
17. Rikitake, K.: *Internet Community*, Ohmsha (1994).