Workgroup: Network Working Group
Internet-Draft:
draft-song-network-aware-dns-01
Published: 19 September 2022
Intended Status: Informational
Expires: 23 March 2023
Authors: H. Song
Futurewei Technologies
D. Eastlake
Futurewei Technologies

The Architecture of Network-Aware Domain Name System (DNS)

Abstract

A simple method of enhancing Domain Name System (DNS) with network awareness is discussed. This enables DNS system responses that are dependent on communication service requirements such as QoS or path without changes in the format of DNS protocol messages or application program interfaces (APIs). The different enhancement methods and use cases are discussed.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 23 March 2023.

Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Different application flows have different requirements on networking, such as bandwidth, delay, jitter, reliability, security, and so on. Many requirements are critical for the quality of service and users are ready for premium services even if extra cost is involved. Meanwhile, today's networks have advanced beyond the best-effort model and are capable of providing per-flow services to meet various application requirements (e.g., QoS) by means of programmability, resource management (e.g., network slicing), traffic engineering, and path regulation (e.g., segment routing and service function chaining).

However, a clear gap exists. Applications usually only care about the abstract requirements ("WHAT") instead of the actual measures for networks to meet such requirements ("HOW"). Therefore, not only is there a lack of a direct means for networks to tell applications their capabilities but also it is often improper to do so. Due to the limitation of the commonly available network socket API, it is also difficult for applications to convey their service requirements to networks. Currently one either assumes the requirements can be expressed to network controllers through some out-of-band manner or, in case of IPv6, by resorting to encoding the requirements as options into extension headers (e.g., network tokens [I-D.yiakoumis-network-tokens]). We need a simpler and more extensible way to set up the service contract.

We define an architecture to support network awareness through DNS. Requirements to network services can be incorporated into DNS queries from a host (e.g., as specified in [I-D.eastlake-dnsop-expressing-qos-requirements]) and the returned information enables access to services meeting those requirements. For example, by

including new semantics representing a service commitment embedded in the returned IP addresses (i.e., semantic addressing [I-D.farrel-irtf-introduction-to-semantic-routing]).

The Domain Name System (DNS) is a distributed database that stores data under hierarchical domain names and supports redundant servers, data caching, and security features. The data is formatted into resource records (RRs) whose content type and structure are indicated by the RR Type field. A typical use of DNS is that, by running the DNS protocol, a host gets the IP addresses stored at a domain name from DNS servers through a DNS resolver. Many other types of data besides IP addresses can be stored in and returned by the DNS.

In a nutshell, the application's service requirements are embedded into the DNS queries from a host. The DNS replies either provide semantic IP addresses or data that help construct the packet header or headers signaling the special packet handling in networks. The application flow packets will use the existing socket API to send the packet. Network devices, after capturing such packets, would decode the semantics and apply any special packet handling accordingly.

This document describes the architecture, requirements, and use cases of the Network-Aware DNS. The details on DNS query encoding and semantic addressing/data in DNS replies will be described in other documents.

1.1. Terminology and Acronyms

The following terminology and acronyms are used in this document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

API - Application Program Interface

DNS - Domain Name System

RR - Resource Record [RFC8499]. The unit of data stored in the DNS.

Semantic Addressing - Encoding extra semantics beyond the destination ID in an address

2. Architecture

The architecture of the Network Aware DNS is shown in Figure 1.

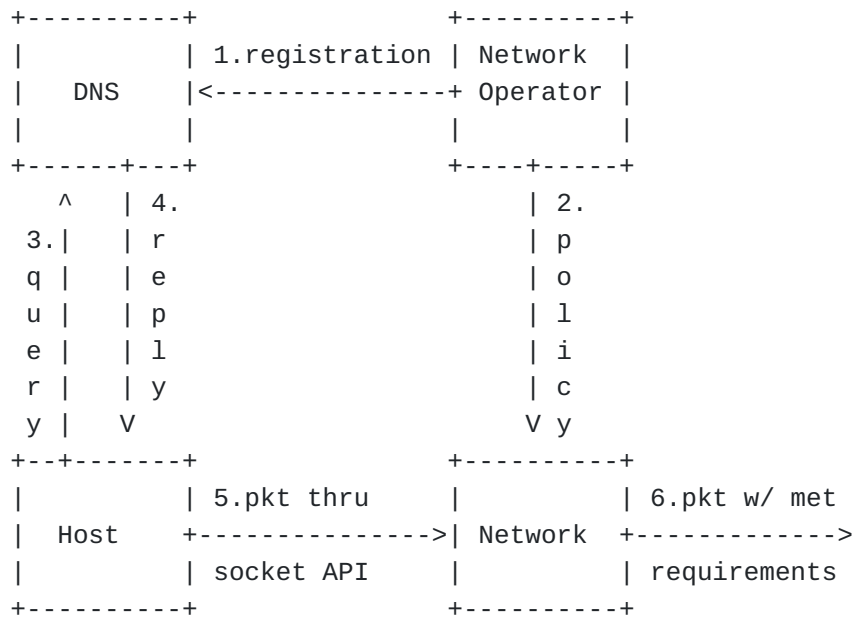

Figure 1: Architecture

1. The network operator registers the semantic addresses/data associated with a name in authoritative DNS servers in form of RRs. In addition to the location, the semantics represent the commitments for network to meet certain service requirements. The semantic addresses or data can be dynamically computed or statically configured by the network operator.

2. Meanwhile, the packet processing policy corresponding to each semantic address/data is configured to the network devices such as routers. How the network meets the service requirements is opaque to host applications.

3. A host application, when conducting a DNS query to a name, would also express its service requirement. A host application can also be ignorant of this service requesting scheme; in this case, the normal DNS query is used and the best-effort results are returned.

4. If the query with service requirements can be satisfied by some RRs in DNS, the result will be returned to the host; otherwise, a normal DNS response, or either an error or the best effort result, will be returned.

5. Once the host application receives the reply, assuming the reply is not an error, it simply uses the address (or assembles the header fields as directed by the semantic data) to forward the packet through a standard socket API. The semantic address or data may be cached at the host for the lifetime of the flow.

Alternatively, the DNS response TTL may indicate the period of time for which the semantic address will provide the service assurances, and the application may again query the DNS at or shortly before the end of the time to refresh the semantic address/data or obtain a new address or data that will be effective for a future interval; however, it is not common for TTL information to be returned to an application doing a DNS query.

6. The network devices would process the packets based on the configured policies if the packets carries semantic addresses and/or header fields. Using a semantic address/data other than for the best effort service might be subject to extra cost based on some service agreement.

We enforce some requirements on the architecture to make it practical for incremental deployment.

*We do not introduce new protocols to enable the architecture.

*As an infrastructural system and protocol, DNS is hard to change. We will not make any change to DNS architecture and protocol. However, within the framework, we have the freedom to introduce new semantics and new RR types to encode semantic data.

*Similarly, it is hard to change the ubiquitous network socket API, so we just rely on it.

*We envision the system would be better used in limited domains where the network operator owns not only the networks but also the proper name servers. In some cases, it is also possible to extend the scope into multiple domains if the packet processing to meet the service requirements can be coordinated cross domains.

*We expect the semantic address or data is per application or per flow based. So each application or flow may need its own DNS name resolution even for the same service. Most applications can still use the conventional best effort service without noticing any change.

In a more dynamic architecture, DNS queries with service requirements can be dynamically sent to the network operator when received by a resolver, allowing network operator to generate on-demand semantic addresses or data for the name server, which will eventually return the information back to the host application.

3. Obtaining Needed Information from DNS

A host application can have three methods to obtain information from the DNS to enable the application to meet its service requirements. These methods are as follows:

Method 1: It sends a requirement-encoded name to ask for an IP address type RR (e.g., AAAA) and expects the semantics to meet the requirements to be embedded in the returned addresses. The encoding method is described in [I-D.eastlake-dnsop-expressing-qos-requirements].

Method 2: It sends a normal name to ask for a different type of RR and the semantic data in the returned RRs represents the means to meet the service requirements.

Method 3: Combining 1 and 2, it sends a requirement-encoded name to ask for a different type of RR, which might be in addition to or lead to (such as the SRV type RR) an IP address type RR, and the semantic data in the RR represents the means to meet the service requirements.

This architecture can support multiple use cases using one of the above methods. Below are some examples.

E2E SRv6: This use case may use method 2. We can support true end-to-end SRv6 service where a Segment List (SL) is acquired from DNS using the RR Type specified in [I-D.eastlake-dnsop-rrtype-srv6] and an SRH (Segment Routing Header) is directly inserted in the IPv6 packet header. While the SRH determines the packet's forwarding path, different packet handling and QoS treatment can also be applied to the packet along the path.

Semantic Addressing: This use case may use method 1. Due to the abundance of IPv6 addresses, each name can be assigned multiple addresses with each representing some special network services. While the network devices are configured or programmed to be able to interpret and process the semantics embedded in addresses, different services can be applied to flows for the same destination. The details are described in a companion draft.

Service Header Fields: This use case may use method 3. Some service-defining header fields (e.g., DSCP in IPv4 header and traffic class and flow label in IPv6 header) can be used to indicate QoS or other service requirements. Such semantic data can also be provided by DNS replies in form of RRs. The details are described in a companion draft.

Other Semantic Data: This use case may apply the method 2 or 3. Some services may have other means to be encapsulated into a

packet (e.g., IPv6 Extension Header). The required information can also be returned by DNS reply as semantic data.

4. Security Considerations

TBD

5. IANA Considerations

This document requires no IANA actions.

6. Acknowledgments

The comments and suggestions of the following are gratefully acknowledged:

*TBD

7. References

7.1. Normative References

[I-D.eastlake-dnsop-expressing-qos-requirements] Eastlake, D. and H. Song, "Expressing Quality of Service Requirements (QoS) in Domain Name System (DNS) Queries", Work in Progress, Internet-Draft, draft-eastlake-dnsop-expressing-qos-requirements-01, 23 August 2022, <https://www.ietf.org/archive/id/draft-eastlake-dnsop-expressing-qos-requirements-01.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[I-D.eastlake-dnsop-rrtype-srv6] Eastlake, D. and H. Song, "An IPv6 Segment Routing (SRv6) Domain Name System (DNS) Resource Record", Work in Progress, Internet-Draft, draft-eastlake-dnsop-rrtype-srv6-01, 30 May 2022, <https://www.ietf.org/archive/id/draft-eastlake-dnsop-rrtype-srv6-01.txt>.

[I-D.farrel-irtf-introduction-to-semantic-routing] Farrel, A. and D. King, "An Introduction to Semantic Routing", Work in

Progress, Internet-Draft, draft-farrel-irtf-introduction-to-semantic-routing-04, 25 April 2022, <https://www.ietf.org/archive/id/draft-farrel-irtf-introduction-to-semantic-routing-04.txt>.

[I-D.yiakoumis-network-tokens] Yiakoumis, Y., McKeown, N., and F. Sorensen, "Network Tokens", Work in Progress, Internet-Draft, draft-yiakoumis-network-tokens-02, 22 December 2020, <https://www.ietf.org/archive/id/draft-yiakoumis-network-tokens-02.txt>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

Authors' Addresses

Haoyu Song
Futurewei Technologies
2220 Central Expressway
Santa Clara, CA 95050
United States of America

Email: haoyu.song@futurewei.com

Donald Eastlake
Futurewei Technologies
2386 Panoramic Circle
Apopka, FL 32703
United States of America

Phone: +1-508-333-2270
Email: d3e3e3@gmail.com