ALTO S. Kiesel
Internet-Draft University of Stuttgart
Intended status: Standards Track M. Stiemerling
Expires: May 18, 2019 H-DA
November 14, 2018

Application Layer Traffic Optimization (ALTO) Cross-Domain Server Discovery
draft-ietf-alto-xdom-disc-04

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications that have to select one or several hosts from a set of candidates capable of providing a desired resource. ALTO is realized by a client-server protocol. Before an ALTO client can ask for guidance it needs to discover one or more ALTO servers that can provide suitable guidance.

In some deployment scenarios, in particular if the information about the network topology is partitioned and distributed over several ALTO servers, it may be needed to discover an ALTO server outside of the own network domain, in order to get appropriate guidance. This document details applicable scenarios, itemizes requirements, and specifies a procedure for ALTO cross-domain server discovery.

Technically, the procedure specified in this document takes one IP address or prefix and a U-NAPTR Service Parameter (typically, "ALTO:https") as parameters. It performs DNS lookups (for NAPTR resource records in the in-addr.arpa. or ip6.arpa. tree) and returns one or more URI(s) of information resources related to that IP address or prefix.

Terminology and Requirements Language

This document makes use of the ALTO terminology defined in RFC 5693 [RFC5693].

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.

Status of this Memo

This Internet-Draft will expire on May 18, 2019.

Copyright Notice

Table of Contents

1. Introduction

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications that have to select one or several hosts from a set of candidates capable of providing a desired resource [RFC5693]. ALTO is realized by an HTTP-based client-server protocol [RFC7285], which can be used in various scenarios [RFC7971].

The ALTO base protocol document [RFC7285] specifies the communication between an ALTO client and one ALTO server. In principle, the client may send any ALTO query. For example, it might ask for the routing cost between any two IP addresses, or it might request network and cost maps for the whole network, which might be the worldwide Internet. It is assumed that the server can answer any query, possibly with some kind of default value if no exact data is known.

No special provisions were made for deployment scenarios with multiple ALTO servers, with some servers having more accurate information about some parts of the network topology while others having better information about other parts of the network ("partitioned knowledge"). Various ALTO use cases have been studied in the context of such scenarios. In some cases, one cannot assume that a topologically nearby ALTO server (e.g., a server discovered with the procedure specified in [RFC7286]) will always provide useful information to the client. One such scenario is detailed in Appendix C. Several solution approaches, such as redirecting a client to a server that has more accurate information or forwarding the request to it on behalf of the client, have been proposed and analyzed (see Appendix A), but none has been specified so far.

Section 3 of this document specifies the "ALTO Cross-Domain Server Discovery Procedure" for client-side usage in these scenarios. An ALTO client that wants to send an ALTO query related to a specific IP address or prefix X, may call this procedure with X as a paramenter. It will use Domain Name System (DNS) lookups to find of one ore more ALTO servers that can provide a competent answer. The above wording "related to" was intentionally kept somewhat vague, as the exact semantics depends on the ALTO service to be used; see Section 4.

Those who are in control of the "reverse DNS" for a given IP address or prefix (i.e., the corresponding subdomain of in-addr.arpa. or ip6.arpa.) - typically an Internet Service Provider (ISP), a corporate IT department, or a university's computing center - may add resource records to the DNS that point to one or more relevant ALTO server(s). In many cases, it may be an ALTO server run by that ISP or IT department, as they naturally have good insight into routing costs from and to their networks. However, they may also refer to an ALTO server provided by someone else, e.g., their upstream ISP.

2. Overview on the ALTO Cross-Domain Server Discovery Procedure

This procedure was inspired by the "Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS" [RFC7216] and re-uses parts of the basic ALTO Server Discovery Procedure [RFC7286].

The basic idea is to use the Domain Name System (DNS), more specifically the "in-addr.arpa." or "ip6.arpa." trees, which are mostly used for "reverse mapping" of IP addresses to host names by means of PTR resource records. There, URI-enabled Naming Authority Pointer (U-NAPTR) resource records [RFC4848], which allow the mapping of domain names to Uniform Resource Identifiers (URIs), are installed as needed. Thereby, it is possible to store a mapping from an IP address or prefix to one or more ALTO server URIs in the DNS.

The ALTO Cross-Domain Server Discovery Procedure is called with one IP address or prefix and a U-NAPTR Service Parameter [RFC4848] as parameters.

The service parameter SHOULD always be set to "ALTO:https". However, other parameter values MAY be used in some scenarios, e.g., "ALTO:http" to search for a server that supports unencrypted transmission for debugging purposes, or other application protocol or service tags if applicable.

The procedure performs DNS lookups and returns one or more URI(s) of information resources related to said IP address or prefix, usually the URI(s) of one or more ALTO Information Resource Directory (IRD, see Section 9 of [RFC7285]). The U-NAPTR records also provide preference values, which should be considered if more than one URI is returned.

The discovery procedure sequentially tries two different lookup strategies: First, an ALTO-specific U-NAPTR record is searched in the "reverse tree", i.e., in subdomains of in-addr.arpa. or ip6.arpa. corresponding to the given IP address or prefix. If this lookup does not yield a usable result, the procedure tries further lookups with truncated domain names, which correspond to shorter prefix lengths. The goal is to allow deployment scenarios that require fine-grained discovery on a per-IP basis, as well as large-scale scenarios where discovery is to be enabled for a large number of IP addresses with a small number of additional DNS resource records.

3. ALTO Cross-Domain Server Discovery Procedure Specification

3.1. Interface

The procedure specified in this document takes two parameters, X and SP, where X is an IP address or prefix and SP is a U-NAPTR Service Parameter.

The parameter X may be an IPv4 or an IPv6 address or prefix in CIDR notation (see [RFC4632] for the IPv4 CIDR notation and [RFC4291] for IPv6). Consequently, the address type AT is either "IPv4" or "IPv6". In both cases, X consists of an IP address A and a prefix length L. For AT=IPv4, it holds: 0 <= L <= 32 and for AT=IPv6, it holds: 0 <= L <= 128.

For example, for X=198.51.100.0/24, we get AT=IPv4, A=198.51.100.0 and L=24. Similarly, for X=2001:0DB8::20/128, we get AT=IPv6, A=2001:0DB8::20 and L=128.

In the intended usage scenario, the procedure SHOULD always be called with the parameter SP set to "ALTO:https". However, for general applicabiliy and in order to support future extensions, the procedure MUST support being called with any valid U-NAPTR Service Parameter (see Section 4.5. of [RFC4848] for the syntax of U-NAPTR Service Parameters and Section 5. of the same document for information about the IANA registries).

The procedure performs DNS lookups and returns one or more URI(s) of information resources related to that IP address or prefix, usually the URI(s) of one or more ALTO Information Resource Directory (IRD, see Section 9 of [RFC7285]). For each URI, it also returns order and preference values (see Section 4.1 of [RFC3403]), which should be considered if more than one URI is returned.

During execution of this procedure, various error conditions may occur and have to be reported to the caller; see Section 3.5.

For the remainder of the document, we use the following notation for calling the ALTO Cross-Domain Server Discovery Procedure:

```
IRD_URIS_X = XDOMDISC(X,"ALTO:https")
```

3.2. Step 1: Prepare Domain Name for Reverse DNS Lookup

First, the procedure checks the prefix length L for unsupported values: If AT=IPv4 (i.e., if A is an IPv4 address) and L < 8, the procedure aborts and indicates an "invalid prefix length" error to the caller. Similarly, if AT=IPv6 and L < 32, the procedure aborts

and indicates an "invalid prefix length" error to the caller. Otherwise, the procedure continues.

If AT=IPv4, a domain name R32 is constructed according to the rules specified in Section 3.5 of [RFC1035] and it is rooted in the special domain "IN-ADDR.ARPA.".

For example, A=198.51.100.3 yields R32="3.100.51.198.IN-ADDR.ARPA.".

If AT=IPv6, the domain name R128 is constructed according to the rules specified in Section 2.5 of [RFC3596] and the special domain "IP6.ARPA." is used.

For example (note: a line break was added after the second line),

```
A = 2001:0DB8::20    yields
R128 = "0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.B.D.0.
        1.0.0.2.IP6.ARPA."
```

3.3. Step 2: Prepare Shortened Domain Names

For this step, an auxiliary function "skip" is defined as follows: skip(str,n) will skip all characters in the string str, up to and including the n-th dot, and return the remaining part of str. For example, skip("foo.bar.baz.qux.quux.",2) will return "baz.qux.quux.".

If AT=IPv4, the following additional domain names are generated from the result of the previous step: R24=skip(R32,1), R16=skip(R32,2), and R8=skip(R32,3). Removing one label from a domain name (i.e., one number of the "dotted quad notation") corresponds to shortening the prefix length by 8 bits.

For example, R32="3.100.51.198.IN-ADDR.ARPA." yields R24="100.51.198.IN-ADDR.ARPA.", R16="51.198.IN-ADDR.ARPA.", and R8="198.IN-ADDR.ARPA.".

If AT=IPv6, the following additional domain names are generated from the result of the previous step: R64=skip(R128,16), R56=skip(R128,18), R48=skip(R128,20), and R32=skip(R128,24). Removing one label from a domain name (i.e., one hex digit) corresponds to shortening the prefix length by 4 bits.

For example (note: a line break was added after the first line),

```
R128 = "0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.B.D.0.
        1.0.0.2.IP6.ARPA."    yields
R64  = "0.0.0.0.0.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.",
R56  = "0.0.0.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.",
R48  = "0.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.", and
R32  = "8.B.D.0.1.0.0.2.IP6.ARPA."
```

3.4. Step 3: Perform DNS U-NAPTR lookups

The address type of A and the prefix length are matched against the first and the second column of the following table, respectively:

1: Address Type AT	2: Prefix Length L	3: MUST do 1st lookup	4: SHOULD do further lookups in that order
IPv4	32	R32	R24, R16, R8
IPv4	24 .. 31	R24	R16, R8
IPv4	16 .. 23	R16	R8
IPv4	8 .. 15	R8	(none)
IPv4	0 .. 7	(none, abort: invalid prefix length L)	
IPv6	128	R128	R64, R56, R48, R32
IPv6	64 (..127)	R64	R56, R48, R32
IPv6	56 .. 63	R56	R48, R32
IPv6	48 .. 55	R48	R32
IPv6	32 .. 47	R32	(none)
IPv6	0 .. 31	(none, abort: invalid prefix length L)	

Then, the domain name given in the 3rd column and the U-NAPTR Service Parameter SP the procedure was called with (usually "ALTO:https") MUST be used for an U-NAPTR [RFC4848] lookup, in order to obtain one or more URIs (indicating protocol, host, and possibly path elements) for the ALTO server's Information Resource Directory (IRD). If such URI(s) can be found, the ALTO Cross-Domain Server Discovery Procedure returns that information to the caller and terminates successfully.

For example, the following two U-NAPTR resource records can be used for mapping "100.51.198.IN-ADDR.ARPA." (i.e., R24 from the example in the previous step) to the HTTPS URIs "https://alto1.example.net/ird" and "https://alto2.example.net/ird", with the former being preferred.

```
100.51.198.IN-ADDR.ARPA.  IN NAPTR 100  10  "u"  "ALTO:https"
     "!.*!https://alto1.example.net/ird!"  ""

100.51.198.IN-ADDR.ARPA.  IN NAPTR 100  20  "u"  "ALTO:https"
     "!.*!https://alto2.example.net/ird!"  ""
```

If no matching U-NAPTR records can be found, the procedure SHOULD try further lookups, using the domain names from the fourth column in the indicated order, until one lookup succeeds. If no IRD URI could be found after looking up all domain names from the 3rd and 4th column, the procedure terminates unsuccessfully, returning an empty URI list.

3.5. Error Handling

The ALTO Cross-Domain Server Discovery Procedure may fail for several reasons.

If the procedure is called with syntactically invalid parameters or unsupported parameter values (in particular the prefix length L, see Section 3.2), the procedure aborts, no URI list will be returned and the error has to be reported to the caller.

The procedure performs one or more DNS lookups in a well-defined order (corresponding to descending prefix lenghts, see Section 3.4), until one produces a usable result. Each of these DNS lookups might not produce a usable result, either due to a normal condition (e.g., domain name exists, but no ALTO-specific NAPTR resource records are associated with it), a permanent error (e.g., non-existent domain name), or due to a temporary error (e.g., timeout). In all three cases, and as long as there are further domain names that can be looked up, the procedure SHOULD immediately try to lookup the next domain name (from column 4 in the table given in Section 3.4). Only after all domain names have been tried at least once, the procedure MAY retry those domain names that had caused temporary lookup errors.

Generally speaking, ALTO provides advisory information for the optimization of applications (e.g., peer-to-peer applications, overlay networks, etc.), but applications should not rely on the availability of such information for their basic functionality (see Section 8.3.4.3 of RFC 7285 [RFC7285]). Consequently, the speedy detection of an ALTO server, even though it may give less accurate answers than other servers, or the quick realization that there is no suitable ALTO server, is in general more preferable than causing long delays by retrying failed queries. Nevertheless, the ALTO Cross-Domain Server Discovery Procedure SHOULD inform its caller, if DNS queries have failed due to temporary errors and that retrying the discovery at a later point in time might give more accurate results.

4. Using the ALTO Protocol with Cross-Domain Server Discovery

Based on a modular design principle, ALTO provides several ALTO services, each consisting of a set of information resouces that can be accessed using the ALTO protocol. The information resources that are available at a specific ALTO server are listed in its Information Resource Directory (IRD, see Section 9 of [RFC7285]). The ALTO protocol specification defines the following ALTO services and their corresponding information resouces:

- o Network and Cost Map Service, see Section 11.2 of [RFC7285]

- o Map-Filtering Service, see Section 11.3 of [RFC7285]

- o Endpoint Property Service, see Section 11.4 of [RFC7285]

- o Endpoint Cost Service, see Section 11.5 of [RFC7285]

The ALTO Cross-Domain Server Discovery Procedure is most useful in conjunction with the Endpoint Property Service and the Endpoint Cost Service. However, for the sake of completeness, possible interaction with all four services is discussed below. Extension documents may specify further information resources; however, these are out of scope of this document.

4.1. Network and Cost Map Service

An ALTO client may invoke the ALTO Cross-Domain Server Discovery Procedure (as specified in Section 3) for an IP address or prefix "X" and get a list of one or more IRD URI(s), including order and preference values: IRD_URIS_X = XDOMDISC(X,"ALTO:https"). These IRD(s) will always contain a network and a cost map, as these are mandatory information resources (see Section 11.2 of [RFC7285]). However, the cost matrix may be very sparse. If, according to the network map, PID_X is the PID that contains the IP address or prefix X, and PID_1, PID_2, PID_3, ... are other PIDS, the cost map may look like this:

```
From \ To PID_1    PID_2    PID_X    PID_3
------+----------------------------------
PID_1 |                        92
PID_2 |                         6
PID_X |      46        3        1       19
PID_3 |                        38
```

In this example, all cells outside column "X" and row "X" are unspecified. A cost map with this structure contains the same information as what could be retrieved using the ECS, cases 1 and 2

in Section 4.4. Accessing cells outside column "X" and row "X" may not yield useful results.

Trying to assemble a more densely populated cost map from several cost maps with this very sparse structure may be a non-trivial task, as different ALTO servers may use different PID definitions (i.e., network maps) and incompatible scales for the costs, in particular for the "routingcost" metric.

4.2. Map-Filtering Service

An ALTO client may invoke the ALTO Cross-Domain Server Discovery Procedure (as specified in Section 3) for an IP address or prefix "X" and get a list of one or more IRD URI(s), including order and preference values: IRD_URIS_X = XDOMDISC(X,"ALTO:https"). These IRD(s) may provide the optional Map-Filtering Service (see Section 11.3 of [RFC7285]). This service returns a subset of the full map, as specified by the client. As discussed in Section 4.1, a cost map may be very sparse in the envisioned deployment scenario. Therefore, depending on the filtering criteria provided by the client, this service may return results similar to the Endpoint Cost Service, or it may not return any useful result.

4.3. Endpoint Property Service

If an ALTO client wants to query an Endpoint Property Service (see Section 11.4 of RFC 7285 [RFC7285]) about an endpoint with IP address "X" or a group of endpoints within IP prefix "X", respectively, it has to invoke the ALTO Cross-Domain Server Discovery Procedure (as specified in Section 3): IRD_URIS_X = XDOMDISC(X,"ALTO:https"). The result IRD_URIS_X is a list of one or more URIs of Information Resource Directories (IRD, see Section 9 of [RFC7285]). Considering the order and preference values, the client has to check these IRDs for a suitable Endpoint Property Service and query it.

If the ALTO client wants to do a similar Endpoint Property query for a different IP address or prefix "Y", the whole procedure has to be repeated, as IRD_URIS_Y = XDOMDISC(Y,"ALTO:https") may yield a different list of IRD URIs. Of course, the results of individual DNS queries may be cached as indicated by their respective time-to-live (TTL) values.

4.4. Endpoint Cost Service

The optional ALTO Endpoint Cost Service (ECS, see Section 11.5 of RFC 7285 [RFC7285]) provides information about costs between individual endpoints and it also supports ranking. The ECS allows that endpoints may be denoted by IP addresses or prefixes. The ECS is

called with a list of one or more source IP addresses or prefixes, which we will call (S1, S2, S3, ...), and a list of one or more destination IP addresses or prefixes, called (D1, D2, D3, ...).

This specification distinguishes several cases, regarding the number of elements in the list of source and destination addresses, respectively:

1. Exactly one source address S1 and more than one destination addresses D1, D2, D3, ... In this case, the ALTO client has to invoke the ALTO Cross-Domain Server Discovery Procedure (as specified in Section 3) with that single source address as a parameter: IRD_URIS_S1 = XDOMDISC(S1,"ALTO:https"). The result IRD_URIS_S1 is a list of one or more URIs of Information Resource Directories (IRD, see Section 9 of [RFC7285]). Considering the order and preference values, the client has to check these IRDs for a suitable Endpoint Cost Service and query it. The ECS is an optional service (see Section 11.5.1 of RFC 7285 [RFC7285]) and therefore, it may well be that an IRD does not refer to an ECS.

   Calling the Cross-Domain Server Discovery Procedure only once with the single source address as a parameter - as opposed to multiple calls, e.g., one for each destination address - is not only a matter of efficiency. In the given scenario, it is advisable to send all ECS queries to the same ALTO server. This ensures that the results can be compared (e.g., for sorting candidate resource providers), even with cost metrics without a well-defined base unit, e.g., the "routingcost" metric.

2. More than one source addresses S1, S2, S3, ... and exactly one destination address D1. In this case, the ALTO client has to invoke the ALTO Cross-Domain Server Discovery Procedure with that single destination address as a parameter: IRD_URIS_D1 = XDOMDISC(D1,"ALTO:https"). The result IRD_URIS_D1 is a list of one or more URIs of IRDs. Considering the order and preference values, the client has to check these IRDs for a suitable ECS and query it.

3. Exactly one source address S1 and exactly one destination address D1. The ALTO client may perform the same steps as in case 1, as specified above. As an alternative, it may also perform the same steps as in case 2, as specified above.

4. More than one source addresses S1, S2, S3, ... and more than one destination addresses D1, D2, D3, ... In this case, the ALTO client should split the list of desired queries based on source addresses and perform separately for each source address the same steps as in case 1, as specified above. As an alternative, the

ALTO client may also group the list based on destination addresses and perform separately for each destination address the same steps as in case 2, as specified above. However, comparing results between these sub-queries may be difficult, in particular if the cost metric is a relative preference without a well-defined base unit (e.g., the "routingcost" metric).

See Appendix C for a detailed example showing the interaction of a tracker-based peer-to-peer application, the ALTO Endpoint Cost Service, and the ALTO Cross-Domain Server Discovery Procedure.

4.5. Summary and Further Extensions

Considering the four services defined in the ALTO base protocol specification [RFC7285], the ALTO Cross-Domain Server Discovery Procedure works best with the Endpoint Property Service (EPS) and the Endpoint Cost Service (ECS). Both the EPS and the ECS take one or more IP addresses as a parameter. The previous sections specify how the parameter for calling the ALTO Cross-Domain Server Discovery Procedure has to be derived from these IP adresses.

In contrast, the ALTO Cross-Domain Server Discovery Procedure seems less useful if the goal is to retrieve network and cost maps that cover the whole network topology. However, the procedure may be useful if a map centered at a specific IP address is desired (i.e., a map detailing the vicinity of said IP address or a map giving costs from said IP address to all potential destinations).

The interaction between further ALTO services (and their corresponding information resources) needs to be investigated and defined once such further ALTO services are specified in an extension document.

5. Implementation, Deployment, and Operational Considerations

5.1. Considerations for ALTO Clients

5.1.1. Resource Consumer Initiated Discovery

To some extent, ALTO requirement AR-32 [RFC6708], i.e., resource consumer initiated ALTO server discovery, can be seen as a special case of cross-domain ALTO server discovery. To that end, an ALTO client embedded in a resource consumer would have to figure out its own "public" IP address and perform the procedures described in this document on that address. However, due to the widespread deployment of Network Address Translators (NAT), additional protocols and mechanisms such as STUN [RFC5389] would be needed and considerations for UNSAF [RFC3424] apply. Therefore, using the procedures specified in this document for resource consumer based ALTO server discovery is generally NOT RECOMMENDED. Note that a less versatile yet simpler approach for resource consumer initiated ALTO server discovery is specified in [RFC7286].

5.1.2. IPv4/v6 Dual Stack, Multihoming, NAT, and Host Mobility

The procedure specified in this document can discover ALTO server URIs for a given IP address or prefix. The intention is, that a third party (e.g., a resource directory) that receives query messages from a resource consumer can use the source address in these messages to discover suitable ALTO servers for this specific resource consumer.

However, resource consumers (as defined in Section 2 of [RFC5693]) may reside on hosts with more than one IP address, e.g., due to IPv4/v6 dual stack operation and/or multihoming. IP packets sent with different source addresses may be subject to different routing policies and path costs. In some deployment scenarios, it may even be required to ask different sets of ALTO servers for guidance. Furthermore, source addresses in IP packets may be modified en-route by Network Address Translators (NAT).

If a resource consumer queries a resource directory for candidate resource providers, the locally selected (and possibly en-route translated) source address of the query message - as observed by the resource directory - will become the basis for the ALTO server discovery and the subsequent optimization of the resource directory's reply. If, however, the resource consumer then selects different source addresses to contact returned resource providers, the desired better-than-random "ALTO effect" may not occur.

Therefore, a dual stack or multihomed resource consumer SHOULD either

always use the same address for contacting the resource directory and the resource providers, i.e., overriding the operating system's automatic source IP address selection, or use resource consumer based ALTO server discovery [RFC7286] to discover suitable ALTO servers for every local address and then locally perform ALTO-influenced resource consumer selection and source address selection. Similarly, resource consumers on mobile hosts SHOULD query the resource directory again after a change of IP address, in order to get a list of candidate resource providers that is optimized for the new IP address.

5.2. Deployment Considerations for Network Operators

5.2.1. Separation of Interests

We assume that if two organizations share parts of their DNS infrastructure, i.e., have common in-addr.arpa. and/or ip6.arpa. subdomains, they will also be able to operate a common ALTO server, which still may do redirections if desired or required by policies.

Note that the ALTO server discovery procedure is supposed to produce only a first URI of an ALTO server that can give reasonable guidance to the client. An ALTO server can still return different results based on the client's address (or other identifying properties) or redirect the client to another ALTO server using mechanisms of the ALTO protocol (see Sect. 9 of [RFC7285]).

6. Security Considerations

A high-level discussion of security issues related to ALTO is part of the ALTO problem statement [RFC5693]. A classification of unwanted information disclosure risks, as well as specific security-related requirements can be found in the ALTO requirements document [RFC6708].

The remainder of this section focuses on security threats and protection mechanisms for the cross-domain ALTO server discovery procedure as such. Once the ALTO server's URI has been discovered and the communication between the ALTO client and the ALTO server starts, the security threats and protection mechanisms discussed in the ALTO protocol specification [RFC7285] apply.

6.1. Integrity of the ALTO Server's URI

Scenario Description
An attacker could compromise the ALTO server discovery procedure or infrastructure in a way that ALTO clients would discover a "wrong" ALTO server URI.

Threat Discussion
This is probably the most serious security concern related to ALTO server discovery. The discovered "wrong" ALTO server might not be able to give guidance to a given ALTO client at all, or it might give suboptimal or forged information. In the latter case, an attacker could try to use ALTO to affect the traffic distribution in the network or the performance of applications (see also Section 15.1. of [RFC7285]). Furthermore, a hostile ALTO server could threaten user privacy (see also Section 5.2.1, case (5a) in [RFC6708]).

However, it should also be noted that, if an attacker was able to compromise the DNS infrastructure used for cross-domain ALTO server discovery, (s)he could also launch significantly more serious other attacks (e.g., redirecting various application protocols).

Protection Strategies and Mechanisms
The cross-domain ALTO server discovery procedure relies on a series of DNS lookups. If an attacker was able to modify or spoof any of the DNS records, the resulting URI could be replaced by a forged URI. The application of DNS security (DNSSEC) [RFC4033] provides a means to limit attacks that rely on modification of the DNS records while in transit. Additional operational precautions for safely operating the DNS infrastructure are required in order to ensure that name servers do not sign forged (or otherwise

"wrong") resource records. Security considerations specific to U-NAPTR are described in more detail in [RFC4848].

A related risk is the impersonation of the ALTO server (i.e., attacks after the correct URI has been discovered). This threat and protection strategies are discussed in Section 15.1 of [RFC7285]. Note that if TLS is used to protect ALTO, the server certificate will contain the host name (CN). Consequently, only the host part of the HTTPS URI will be authenticated, i.e., the result of the ALTO server discovery procedure. The DNS/U-NAPTR based mapping within the cross-domain ALTO server discovery procedure needs to be secured as described above, e.g., by using DNSSEC.

In addition to active protection mechanisms, users and network operators can monitor application performance and network traffic patterns for poor performance or abnormalities. If it turns out that relying on the guidance of a specific ALTO server does not result in better-than-random results, the usage of the ALTO server may be discontinued (see also Section 15.2 of [RFC7285]).

6.2. Availability of the ALTO Server Discovery Procedure

Scenario Description

An attacker could compromise the cross-domain ALTO server discovery procedure or infrastructure in a way that ALTO clients would not be able to discover any ALTO server.

Threat Discussion

If no ALTO server can be discovered (although a suitable one exists) applications have to make their decisions without ALTO guidance. As ALTO could be temporarily unavailable for many reasons, applications must be prepared to do so. However, The resulting application performance and traffic distribution will correspond to a deployment scenario without ALTO.

Protection Strategies and Mechanisms

Operators should follow best current practices to secure their DNS and ALTO (see Section 15.5 of [RFC7285]) servers against Denial-of-Service (DoS) attacks.

6.3. Confidentiality of the ALTO Server's URI

Scenario Description

An unauthorized party could invoke the cross-domain ALTO server discovery procedure, or intercept discovery messages between an authorized ALTO client and the DNS servers, in order to acquire knowledge of the ALTO server URI for a specific IP address.

Threat Discussion

In the ALTO use cases that have been described in the ALTO problem statement [RFC5693] and/or discussed in the ALTO working group, the ALTO server's URI as such has always been considered as public information that does not need protection of confidentiality.

Protection Strategies and Mechanisms

No protection mechanisms for this scenario have been provided, as it has not been identified as a relevant threat. However, if a new use case is identified that requires this kind of protection, the suitability of this ALTO server discovery procedure as well as possible security extensions have to be re-evaluated thoroughly.

6.4. Privacy for ALTO Clients

Scenario Description

An unauthorized party could intercept messages between an ALTO client and the DNS servers, and thereby find out the fact that said ALTO client uses (or at least tries to use) the ALTO service in order to optimize traffic from/to a specific IP address.

Threat Discussion

In the ALTO use cases that have been described in the ALTO problem statement [RFC5693] and/or discussed in the ALTO working group, this scenario has not been identified as a relevant threat.

Protection Strategies and Mechanisms

No protection mechanisms for this scenario have been provided, as it has not been identified as a relevant threat. However, if a new use case is identified that requires this kind of protection, the suitability of this ALTO server discovery procedure as well as possible security extensions have to be re-evaluated thoroughly.

7. IANA Considerations

This document does not require any IANA action.

8. References

8.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3403] Mealling, M., "Dynamic Delegation Discovery System (DDDS) Part Three: The Domain Name System (DNS) Database", RFC 3403, October 2002.

[RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, October 2003.

[RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, April 2007.

[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>.

8.2. Informative References

[I-D.kiesel-alto-alto4alto]
Kiesel, S., "Using ALTO for ALTO server selection", draft-kiesel-alto-alto4alto-00 (work in progress), July 2010.

[I-D.kiesel-alto-ip-based-srv-disc]
Kiesel, S. and R. Penno, "Application-Layer Traffic Optimization (ALTO) Anycast Address", draft-kiesel-alto-ip-based-srv-disc-03 (work in progress), July 2014.

[RFC3424] Daigle, L. and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing

Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <http://www.rfc-editor.org/info/rfc4632>.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

[RFC6708] Kiesel, S., Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", RFC 6708, September 2012.

[RFC7216] Thomson, M. and R. Bellis, "Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS", RFC 7216, April 2014.

[RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

[RFC7286] Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and H. Song, "Application-Layer Traffic Optimization (ALTO) Server Discovery", RFC 7286, June 2014.

[RFC7971] Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "Application-Layer Traffic Optimization (ALTO) Deployment Considerations", RFC 7971, DOI 10.17487/RFC7971, October 2016, <http://www.rfc-editor.org/info/rfc7971>.

Appendix A. Solution Approaches for Partitioned ALTO Knowledge

The ALTO base protocol document [RFC7285] specifies the communication between an ALTO client and a single ALTO server. It is implicitly assumed that this server can answer any query, possibly with some kind of default value if no exact data is known. No special provisions were made for the case that the ALTO information originates from multiple sources, which are possibly under the control of different administrative entities (e.g., different ISPs) or that the overall ALTO information is partitioned and stored on several ALTO servers.

A.1. Classification of Solution Approaches

Various protocol extensions and other solutions have been proposed to deal with multiple information sources and partitioned knowledge. They can be classified as follows:

1   Ensure that all ALTO servers have the same knowledge

1.1 Ensure data replication and synchronization within the provisioning protocol (cf. RFC 5693, Fig 1 [RFC5693]).

1.2 Use an Inter-ALTO-server data replication protocol. Possibly, the ALTO protocol itself - maybe with some extensions - could be used for that purpose; however, this has not been studied in detail so far.

2   Accept that different ALTO servers (possibly operated by different organizations, e.g., ISPs) do not have the same knowledge

2.1 Allow ALTO clients to send arbitrary queries to any ALTO server (e.g. the one discovered using [RFC7286]). If this server cannot answer the query itself, it will fetch the data on behalf of the client, using the ALTO protocol or a to-be-defined inter-ALTO-server request forwarding protocol.

2.2 Allow ALTO clients to send arbitrary queries to any ALTO server (e.g. the one discovered using [RFC7286]). If this server cannot answer the query itself, it will redirect the client to the "right" ALTO server that has the desired information, using a small to-be-defined extension of the ALTO protocol.

2.3 ALTO clients need to use some kind of "search engine" that indexes ALTO servers and redirects and/or gives cached results.

2.4 ALTO clients need to use a new discovery mechanism to discover the ALTO server that has the desired information and contact it directly.

A.2. Discussion of Solution Approaches

The provisioning or initialization protocol for ALTO servers (cf. RFC 5693, Fig 1 [RFC5693]) is currently not standardized. It was a conscious decision not to include this in the scope of the IETF ALTO working group. The reason is that there are many different kinds of information sources. This implementation specific protocol will adapt them to the ALTO server, which offers a standardized protocol to the ALTO clients. However, adding the task of synchronization between ALTO servers to this protocol (i.e., approach 1.1) would overload this protocol with a second functionality that requires standardization for seamless multi-domain operation.

For the 1.? solution approaches, in addition to general technical feasibility and issues like overhead and caching efficiency, another aspect to consider is legal liability. Operator "A" might prefer not to publish information about nodes in or paths between the networks of operators "B" and "C" through A's ALTO server, even if A knew that information. This is not only a question of map size and processing load on A's ALTO server. Operator A could also face legal liability issues if that information had a bad impact on the traffic engineering between B's and C's networks, or on their business models.

No specific actions to build a "search engine" based solution (approach 2.3) are currently known and it is unclear what could be the incentives to operate such an engine. Therefore, this approach is not considered in the remainder of this document.

A.3. The Need for Cross-Domain ALTO Server Discovery

Approaches 1.1, 1.2, 2.1, and 2.2 do not only require the specification of an ALTO protocol extension or a new protocol that runs between ALTO servers. A large-scale, maybe Internet-wide, multi-domain deployment would also need mechanisms by which an ALTO server could discover other ALTO servers, learn which information is available where, and ideally also who is authorized to publish information related to a given part of the network. Approach 2.4 needs the same mechanisms, except that they are used on the client-side instead of the server-side.

It is sometimes questioned whether there is a need for a solution that allows clients to ask arbitrary queries, even if the ALTO information is partitioned and stored on many ALTO servers. The main

argument is, that clients are supposed to optimize the traffic from and to themselves, and that the information needed for that is most likely stored on a "nearby" ALTO server, i.e., the one that can be discovered using [RFC7286]. However, there are scenarios where the ALTO client is not co-located with an endpoint of the to-be-optimized data transmission. Instead, the ALTO client is located at a third party, which takes part in the application signaling, e.g., a so-called "tracker" in a peer-to-peer application. One such scenario, where it is advantageous to place the ALTO client not at an endpoint of the user data transmission, is analyzed in Appendix C.

A.4. Our Solution Approach

Several solution approaches for cross-domain ALTO server discovery have been evaluated, using the criteria documented in Appendix B. One of them was to use the ALTO protocol itself for the exchange of information availability [I-D.kiesel-alto-alto4alto]. However, the drawback of that approach is that a new registration administration authority would have to be established.

This document specifies a DNS-based procedure for cross-domain ALTO server discovery, which was inspired by "Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS" [RFC7216]. The primary goal is that this procedure can be used on the client-side (i.e., approach 2.4), but together with new protocols or protocol extensions it could also be used to implement the other solution approaches itemized above.

A.5. Relation to the ALTO Requirements

During the design phase of the overall ALTO solution, two different server discovery scenarios have been identified and documented in the ALTO requirements document [RFC6708]. The first scenario, documented in Req. AR-32, can be supported using the discovery mechanisms specified in [RFC7286]. An alternative approach, based on IP anycast [I-D.kiesel-alto-ip-based-srv-disc], has also been studied. This document, in contrast, tries to address Req. AR-33.

Appendix B. Requirements for ALTO Cross-Domain Server Discovery

This appendix itemizes requirements that have been collected before the design phase and that are reflected by the design of the ALTO Cross-Domain Server Discovery Procedure.

B.1. Discovery Client Application Programming Interface

The discovery client will be called through some kind of application programming interface (API) and the parameters will be an IP address and, for purposes of extensibility, a service identifier such as "ALTO". It will return one or more URI(s) that offers the requested service ("ALTO") for the given IP address.

In other words, the client would be used to retrieve a mapping:

(IP address, "ALTO") -> IRD-URI(s)

where IRD-URI(s) is one or more URI(s) of Information Resource Directories (IRD, see Section 9 of [RFC7285]) of ALTO server(s) that can give reasonable guidance to a resource consumer with the indicated IP address.

B.2. Data Storage and Authority Requirements

The information for mapping IP addresses and service parameters to URIs should be stored in a - preferably distributed - database. It must be possible to delegate administration of parts of this database. Usually, the mapping from a specific IP address to an URI is defined by the authority that has administrative control over this IP address, e.g., the ISP in residential access networks or the IT department in enterprise, university, or similar networks.

B.3. Cross-Domain Operations Requirements

The cross-domain server discovery mechanism should be designed in such a way that it works across the public Internet and also in other IP-based networks. This in turn means that such mechanisms cannot rely on protocols that are not widely deployed across the Internet or protocols that require special handling within participating networks. An example is multicast, which is not generally available across the Internet.

The ALTO Cross-Domain Server Discovery protocol must support gradual deployment without a network-wide flag day. If the mechanism needs some kind of well-known "rendezvous point", re-using an existing infrastructure (such as the DNS root servers or the WHOIS database) should be preferred over establishing a new one.

B.4. Protocol Requirements

The protocol must be able to operate across middleboxes, especially across NATs and firewalls.

The protocol shall not require any pre-knowledge from the client other than any information that is known to a regular IP host on the Internet.

B.5. Further Requirements

The ALTO cross domain server discovery cannot assume that the server discovery client and the server discovery responding entity are under the same administrative control.

Appendix C. ALTO and Tracker-based Peer-to-Peer Applications

This appendix illustrates one ALTO use case and shows that ALTO Cross-Domain Server Discovery is beneficial in that scenario.

C.1. Architectural Options

The ALTO protocol specification [RFC7285] details how an ALTO client can query an ALTO server for guiding information and receive the corresponding replies. However, in the considered scenario of a tracker-based P2P application, there are two fundamentally different possibilities where to place the ALTO client:

1. ALTO client in the resource consumer ("peer")

2. ALTO client in the resource directory ("tracker")

In the following, both scenarios are compared in order to explain the need for ALTO queries on behalf of remote resource consumers.

In the first scenario (see Figure 2), the resource consumer queries the resource directory for the desired resource (F1). The resource directory returns a list of potential resource providers without considering ALTO (F2). It is then the duty of the resource consumer to invoke ALTO (F3/F4), in order to solicit guidance regarding this list.

In the second scenario (see Figure 4), the resource directory has an embedded ALTO client. After receiving a query for a given resource (F1) the resource directory invokes this ALTO client to evaluate all resource providers it knows (F2/F3). Then it returns a, possibly shortened, list containing the "best" resource providers to the resource consumer (F4).

```
    .............................          .............................
    : Tracker                   :          : Peer                      :
    :   ______                  :          :                           :
    : +-______-+                :          :            k good         :
    : |        |     +--------+ : P2P App. : +--------+ peers +------+ :
    : |   N    |     | random | : Protocol : | ALTO-  |------>| data | :
    : | known  |====>| pre-   |*************>| biased |       | ex-  | :
    : | peers, |     | selec- | : transmit : | peer   |------>| cha- | :
    : | M good |     | tion   | : n peer   : | select | n-k   | nge  | :
    : +-______-+     +--------+ : IDs      : +--------+ bad p.+------+ :
    :...........................:          :.....^.....................:
                                                 |
                                                 | ALTO
                                                 | client protocol
                                               __|___
                                             +-______-+
                                             |        |
                                             | ALTO   |
                                             | server |
                                             +-______-+
```

Figure 1: Tracker-based P2P Application with random peer preselection

```
   Peer w. ALTO cli.            Tracker               ALTO Server
   --------+--------       --------+--------       --------+--------
           | F1 Tracker query      |                       |
           |=====================>|                       |
           | F2 Tracker reply      |                       |
           |<=====================|                       |
           | F3 ALTO client protocol query                 |
           |---------------------------------------------->|
           | F4 ALTO client protocol reply                 |
           |<----------------------------------------------|
           |                       |                       |

   ====  Application protocol (i.e., tracker-based P2P app protocol)
   ----  ALTO client protocol
```

Figure 2: Basic message sequence chart for resource consumer-initiated ALTO query

```
    .............................          .............................
    : Tracker                    :          : Peer                       :
    :   ______                   :          :                            :
    : +-______-+                 :          :                            :
    : |        |     +--------+ : P2P App. :  k good peers &  +------+ :
    : |   N    |     | ALTO-   | : Protocol :  n-k bad peers   | data | :
    : | known  |====>| biased  |******************************>| ex-  | :
    : | peers, |     | peer    | : transmit :                  | cha- | :
    : | M good |     | select  | : n peer   :                  | nge  | :
    : +-______-+     +--------+ : IDs      :                  +------+ :
    :.....................^......:          :............................:
                          |
                          | ALTO
                          | client protocol
                        __|___
                      +-______-+
                      |        |
                      | ALTO   |
                      | server |
                      +-______-+
```

Figure 3: Tracker-based P2P Application with ALTO client in tracker

```
        Peer             Tracker w. ALTO cli.       ALTO Server
   --------+--------       --------+--------       --------+--------
           | F1 Tracker query      |                       |
           |======================>|                       |
           |                       | F2 ALTO cli. p. query |
           |                       |---------------------->|
           |                       | F3 ALTO cli. p. reply |
           |                       |<----------------------|
           | F4 Tracker reply      |                       |
           |<======================|                       |
           |                       |                       |

   ====  Application protocol (i.e., tracker-based P2P app protocol)
   ----  ALTO client protocol
```

Figure 4: Basic message sequence chart for ALTO query on behalf of remote resource consumer

Note: the message sequences depicted in Figure 2 and Figure 4 may occur both in the target-aware and the target-independent query mode (c.f. [RFC6708]). In the target-independent query mode no message exchange with the ALTO server might be needed after the tracker query, because the candidate resource providers could be evaluated using a locally cached "map", which has been retrieved from the ALTO

server some time ago.

C.2. Evaluation

The problem with the first approach is, that while the resource directory might know thousands of peers taking part in a swarm, the list returned to the resource consumer is usually shortened for efficiency reasons. Therefore, the "best" (in the sense of ALTO) potential resource providers might not be contained in that list anymore, even before ALTO can consider them.

For illustration, consider a simple model of a swarm, in which all peers fall into one of only two categories: assume that there are "good" ("good" in the sense of ALTO's better-than-random peer selection, based on an arbitrary desired rating criterion) and "bad' peers only. Having more different categories makes the maths more complex but does not change anything to the basic outcome of this analysis. Assume that the swarm has a total number of N peers, out of which are M "good" and N-M "bad" peers, which are all known to the tracker. A new peer wants to join the swarm and therefore asks the tracker for a list of peers.

If, according to the first approach, the tracker randomly picks n peers from the N known peers, the result can be described with the hypergeometric distribution. The probability that the tracker reply contains exactly k "good" peers (and n-k "bad" peers) is:

$$P(X=k) = \frac{\binom{M}{k}\binom{N-M}{n-k}}{\binom{N}{n}}$$

$$\text{with} \quad \binom{n}{k} = \frac{n!}{k!\,(n-k)!} \quad \text{and} \quad n! = n * (n-1) * (n-2) * .. * 1$$

The probability that the reply contains at most k "good" peers is: P(X<=k)=P(X=0)+P(X=1)+..+P(X=k).

For example, consider a swarm with N=10,000 peers known to the tracker, out of which M=100 are "good" peers. If the tracker randomly selects n=100 peers, the formula yields for the reply: P(X=0)=36%, P(X<=4)=99%. That is, with a probability of approx. 36%

this list does not contain a single "good" peer, and with 99% probability there are only four or less of the "good" peers on the list. Processing this list with the guiding ALTO information will ensure that the few favorable peers are ranked to the top of the list; however, the benefit is rather limited as the number of favorable peers in the list is just too small.

Much better traffic optimization could be achieved if the tracker would evaluate all known peers using ALTO, and return a list of 100 peers afterwards. This list would then include a significantly higher fraction of "good" peers. (Note, that if the tracker returned "good" peers only, there might be a risk that the swarm might disconnect and split into several disjunct partitions. However, finding the right mix of ALTO-biased and random peer selection is out of the scope of this document.)

Therefore, from an overall optimization perspective, the second scenario with the ALTO client embedded in the resource directory is advantageous, because it is ensured that the addresses of the "best" resource providers are actually delivered to the resource consumer. An architectural implication of this insight is that the ALTO server discovery procedures must support ALTO queries on behalf of remote resource consumers. That is, as the tracker issues ALTO queries on behalf of the peer which contacted the tracker, the tracker must be able to discover an ALTO server that can give guidance suitable for that respective peer. This task can be solved using the ALTO Cross-Domain Server Discovery Procedure.

C.3. Example

This section provides a complete example of the ALTO Cross-Domain Server Discovery Procedure in a tracker-based peer-to-peer scenario.

The example is based on the network topology shown in Figure 5. Five access networks - Networks a, b, c, x, and t - are operated by five different network operators. They are interconnected by a backbone structure. Each network operator runs an ALTO server in their network, i.e., ALTO_SRV_A, ALTO_SRV_B, ALTO_SRV_C, ALTO_SRV_X, and ALTO_SRV_T, respectively.

```
        _____    __             _____    __             _____    __
     __(     )__(  )_        __(     )__(  )_        __(     )__(  )_
    (   Network a    )      (   Network b    )      (   Network c    )
   ( Res. Provider A  )    ( Res. Provider B  )    ( Res. Provider C  )
    (__ ALTO_SRV_A __)      (__ ALTO_SRV_B __)      (__ ALTO_SRV_C __)
      (___)--(____) \         (___)--(____)         / (___)--(____)
                     \           /                 /
                   ---+---------+-----------------+----
                  (              Backbone              )
                   ------------+-----------------+----
                   _____    __/              _____   \__
                __(     )__(  )_          __(     )__(  )_
               (   Network x    )        (   Network t    )
              ( Res. Consumer X  )      (Resource Directory)
               (_  ALTO_SRV_X __)        (_  ALTO_SRV_T __)
                 (___)--(____)             (___)--(____)
```

Figure 5: Example network topology

A new peer of a peer-to-peer application wants to join a specific swarm (overlay network), in order to access a specific resource. This new peer will be called "Resource Consumer X" in accordance to the terminology of [RFC6708] and it is located in Network x. It contacts the tracker ("Resource Directory"), which is located in Network t. The mechanism by which the new peer discovers the tracker is out of the scope of this document. The tracker maintains a list of peers that take part in the overlay network, and hence it can determine that Resource Providers A, B, and C are candidate peers for Resource Consumer X.

As shown in the previous section, a tracker-side ALTO optimization (c.f. Figure 3 and Figure 4) is more efficient than a client-side optimization. Consequently, the tracker wants to use the ALTO Endpoint Cost Service (ECS) to learn the routing costs between X and A, X and B, as well as X and C, in order to sort A, B, and C by their respective routing costs to X.

In theory, there are many options how the ALTO Cross-Domain Server Discovery Procedure could be used. For example, the tracker could do the following steps:

```
IRD_URIS_A = XDOMDISC(A,"ALTO:https")
COST_X_A   = query the ECS(X,A,routingcost) found in IRD_URIS_A

IRD_URIS_B = XDOMDISC(B,"ALTO:https")
COST_X_B   = query the ECS(X,B,routingcost) found in IRD_URIS_B

IRD_URIS_C = XDOMDISC(C,"ALTO:https")
COST_X_C   = query the ECS(X,C,routingcost) found in IRD_URIS_C
```

Maybe, the ALTO Cross-Domain Server Discovery Procedure queries would yield in this scenario: IRD_URIS_A = ALTO_SRV_A, IRD_URIS_B = ALTO_SRV_B, and IRD_URIS_C = ALTO_SRV_C. That is, each ECS query would be sent to a different ALTO server. The problem with this approach is that we are not neccessarily able to compare COST_X_A, COST_X_B, and COST_X_C with each other. The specification of the routingcost metric mandates that "A lower value indicates a higher preference", but "an ISP may internally compute routing cost using any method that it chooses" (see section 6.1.1.1. of [RFC7285]). Thus, COST_X_A could be 10 (milliseconds round-trip time), while COST_X_B could be 200 (kilometers great circle distance between the approximate geographic locations of the hosts) and COST_X_C could be 3 (router hops, corresponding to a decrease of the TTL field in the IP header). Each of these metrics fulfills the "lower value is more preferable" requirement on its own, but obviously, they cannot be compared with each other. Even if there was a reasonable formula to compare, for example, kilometers with milliseconds, we could not use it, as the units of measurement (or any other information about the computation method for the routingcost) are not sent along with the value in the ECS reply.

To avoid this problem, the tracker tries to send all ECS queries to the same ALTO server. As specified in Section 4.4 of this document, case 2, it uses the IP address of Resource Consumer x as parameter to the discovery procedure:

```
IRD_URIS_X = XDOMDISC(X,"ALTO:https")
COST_X_A   = query the ECS(X,A,routingcost) found in IRD_URIS_X
COST_X_B   = query the ECS(X,B,routingcost) found in IRD_URIS_X
COST_X_C   = query the ECS(X,C,routingcost) found in IRD_URIS_X
```

This strategy ensures that COST_X_A, COST_X_B, and COST_X_C can be compared with each other.

As discussed above, the tracker calls the ALTO Cross-Domain Server Discovery Procedure with IP address X as a parameter. For the reminder of this example, we assume that X = 2001:DB8:1:2:227:eff:fe6a:de42. Thus, the procedure call is

IRD_URIS_X = XDOMDISC(2001:DB8:1:2:227:eff:fe6a:de42,"ALTO:https").

The first parameter 2001:DB8:1:2:227:eff:fe6a:de42 is a single IPv6 address. Thus, we get AT = IPv6, A = 2001:DB8:1:2:227:eff:fe6a:de42, L = 128, and SP = "ALTO:https".

The procedure constructs (see Step 1 in Section 3.2)
R128 = "2.4.E.D.A.6.E.F.F.F.E.0.7.2.2.0.2.0.0.0.1.0.0.0.
8.B.D.0.1.0.0.2.IP6.ARPA.", as well as (see Step 2 in Section 3.3)
R64 = "2.0.0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.",
R56 = "0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.",
R48 = "1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.", and
R32 = "8.B.D.0.1.0.0.2.IP6.ARPA.".

In order to illustrate the third step of the ALTO Cross-Domain Server Discovery Procedure, we use the "dig" (domain information groper) DNS lookup utility that is available for many operating systems (e.g., Linux). A real implementation of the ALTO Cross-Domain Server Discovery Procedure would not be based on the "dig" utility, but use appropriate libraries and/or operating system APIs. Please note that the following steps have been performed in a controlled lab environment with a appropriately configured name server. A suitable DNS configuration will be needed to reproduce these results. Please also note that the rather verbose output of the "dig" tool has been shortened to the relevant lines.

Since AT = IPv6 and L = 128, in the table given in Section 3.4, the sixth row (not counting the column headers) applies.

As mandated by the third column, we start with a lookup of R128, looking for NAPTR resource records:

```
| user@labpc:~$ dig -tNAPTR 2.4.E.D.A.6.E.F.F.F.E.0.7.2.2.0.\
| 2.0.0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.
|
| ;; Got answer:
| ;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 26553
| ;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADD'L: 0
```

The domain name R128 does not exist (status: NXDOMAIN), so we cannot get a useful result. Therefore, we continue with the fourth column of the table and do a lookup of R64:

```
| user@labpc:~$ dig -tNAPTR 2.0.0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.
|
| ;; Got answer:
| ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 33193
| ;; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADD'L: 0
```

The domain name R64 could be looked up (status: NOERROR), but there are no NAPTR resource records associated with it (ANSWER: 0). Maybe, there are some other resource records such as PTR, NS, or SOA, but we are not interested in them. Thus, we do not get a useful result and we continue with looking up R56:

```
| user@labpc:~$ dig -tNAPTR 0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.
|
| ;; Got answer:
| ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35966
| ;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADD'L: 2
|
| ;; ANSWER SECTION:
| 0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA. 604800 IN NAPTR 100 10 "u"
|  "LIS:HELD" "!.*!https://lis1.example.org:4802/?c=ex!" .
| 0.0.1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA. 604800 IN NAPTR 100 20 "u"
|  "LIS:HELD" "!.*!https://lis2.example.org:4802/?c=ex!" .
```

The domain name R56 could be looked up and there are NAPTR resource records associated with it. However, each of these records has a service parameter that does not match our SP = "ALTO:https" (see [RFC7216] for "LIS:HELD"), and therefore, we have to ignore them. Consequently, we still do not have a useful result and continue with a lookup of R48:

```
| user@labpc:~$ dig -tNAPTR 1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA.
|
| ;; Got answer:
| ;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50459
| ;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 1, ADD'L: 2
|
| ;; ANSWER SECTION:
| 1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA. 604800 IN NAPTR 100 10 "u"
|  "ALTO:https" "!.*!https://alto1.example.net/ird!" .
| 1.0.0.0.8.B.D.0.1.0.0.2.IP6.ARPA. 604800 IN NAPTR 100 10 "u"
|  "LIS:HELD" "!.*!https://lis.example.net:4802/?c=ex!" .
```

This lookup yields two NAPTR resource records. We have to ignore the second one as its service parameter does not match our SP, but the first NAPTR resource record has a matching service parameter. Therefore, the procedure terminates successfully and the final outcome is: IRD_URIS_X = "https://alto1.example.net/ird".

The ALTO client that is embedded in the tracker will access the ALTO Information Resource Directory (IRD, see Section 9 of [RFC7285]) at this URI, look for the Endpoint Cost Service (ECS, see Section 11.5 of [RFC7285]), and query the ECS for the costs between A and X, B and X, as well as C and X, before returning an ALTO-optimized list of candidate resource providers to resource consumer X.

Appendix D. Contributors List and Acknowledgments

The initial version of this document was co-authored by Marco Tomsu (Alcatel-Lucent).

This document borrows some text from [RFC7286], as historically, it has been part of the draft that eventually became said RFC. Special thanks to Michael Scharf and Nico Schwan.

Authors' Addresses

Sebastian Kiesel
University of Stuttgart Information Center
Allmandring 30
Stuttgart 70550
Germany

Email: ietf-alto@skiesel.de
URI: http://www.izus.uni-stuttgart.de

Martin Stiemerling
University of Applied Sciences Darmstadt, Computer Science Dept.
Haardtring 100
Darmstadt 64295
Germany

Phone: +49 6151 16 37938
Email: mls.ietf@gmail.com
URI: http://ietf.stiemerling.org