alto-queryresponse-00

Saumitra Das

Peer input not enough

- Goal
  - Optimize data transfer performance
- Leads to
  - Traffic crosses network boundaries multiple times

ISP input is not enough

- Some of the information needed may not be available to the ISP

- ISP's goals are to
  - Minimize congestion near hotspots
  - Minimize interdomain traffic

- That leads to
  - ISP cost reduction not always tied to improved application performance

- All hosts within the ISP are DSL or overloaded
- Geographically close peers with different access network

Goals

- Provide applications with information to perform better-than-random peer selection for different contexts
  - Picking a peer for content download
  - Neighbor selection
  - Picking a super peer, mirror, TURN server, etc.
- The information needed to assist peer selection is likely quite different for different contexts
  - bandwidth, network coordinate, services offered, network impact, cost, etc.
- However, a common framework for such information exchange is very feasible

Entities in the ALTO system

- ALTO Service Providers
  - Network operators, third-parties, communities
- ALTO clients
  - handsets, desktops, netbooks, gaming devices etc
- ALTO aggregators
  - can aggregate information and speed up responses to clients

Assumption

The protocol operates over a single given interface at a time. The whole procedure can be repeated for a different interface. Each interface on a multihomed host may require the discovery of a different ALTO service (since the ISPs on each interface may be different). The peer selection is also dependent on the interface. Hence the protocol is meant to operate per interface. Any peer selection algorithms that work across interfaces would need to perform ALTO query reponse on each interface and use its own algorithm to decide which peer to connect to on which interface.

Discovery

- Basic goal is to get a ALTO service configuration document from an ALTO server
- Use out of band mechanisms or DNS SRV query to determine ALTO server address
- OPEN ISSUE: performing discovery in the context of an overlay. Maybe similar to TURN server discovery in p2psip.

Configuration document

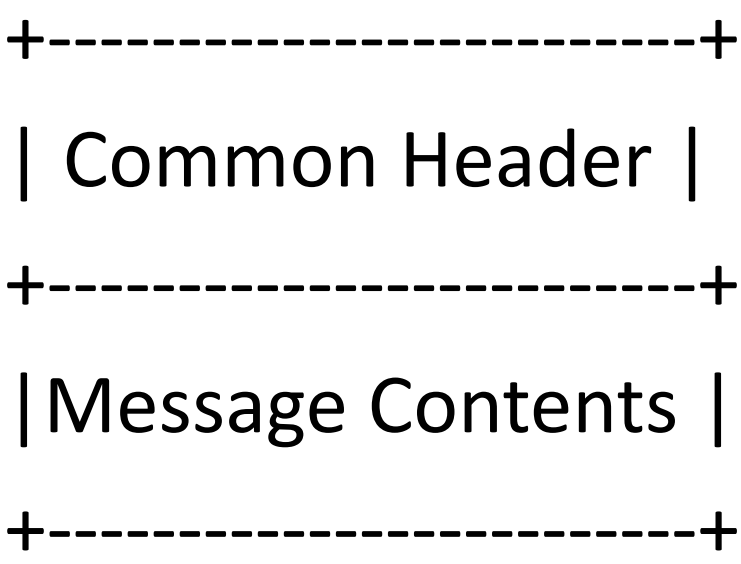
```
<?xml version="1.0" encoding="UTF-8"?>
    <alto instance-name="alto.isp.com"
   expiration="86400">
      <alto-server uri="http://192.168.1.19:75"/>
      <location-context value="false"/>
      <metric-supported name="latency"/>
      <metric-supported name="pDistance"/>
      <metric-supported name="bandwidth"/>
    </alto>
```

Configuration Document

- OPEN ISSUE: Which metrics should be standardized and what units should be used

- TODO: Sync these metrics with what is defined for p2psip diagnostics. The metrics used in p2psip diagnostics may be metrics that are useful to aggregate at an ALTO service in an overlay. This service can then provide ALTO clients with peer selection guidance.

Message Formats



Common Header

```
struct {
    uint8 version;
    uint24 length;
    CommonOptions options[options_length];
} CommonHeader;

enum { (255) } CommonOptionsType;
struct {
    CommonOptionsType type;
    uint8 flags;
    uint16 length;
    select (type) { /* Option values go here */ } option;
} CommonOption;
```

Message contents

```
struct {
    MessageCode message_code;
    opaque payload<0..2^24-1>;
} MessageContents;
```

Location Query and Response

```
struct {
    uint8 precision;
} LocationQuery;

struct {
        HostLocationAttribute hla;
} Response;
```

Host Location Attribute

```
enum {reserved_addr(0), ipv4_address (1), ipv6_address (2), node_address (3), partition_id(4),
        (255)} HostLocationAttributeType;

    typedef opaque  OverlayNodeAddr[16];

    typedef opaque  ISPPartitionAddr[16];

    struct {
      AddressType           type;
      uint8                 length;

      select (type) {
        case ipv4_address:
          IPv4Addr      v4addr;
        case ipv6_address:
          IPv6Addr      v6addr;
        case node_address:
          OverlayNodeAddr       nodeaddr;
        case partition:
          ISPPartitionAddr      partitionaddr;
        /* This structure can be extended */
      } HostLocationAttribute;
```

Guidance Query

```
struct{
      uint8 metricname;
   uint8 operator;
   uint16 value;
}MetricData;

struct {
      ObjectId oid;
      HostLocationAttribute clienthla;
      uint8 precision;
      MetricData metrics<0..2^32-1>
       HostLocationAttribute destinations<0..2^32-1>;
} Query;
```

Guidance Query

- clienthla: The host location attribute of the client. This need not be the same as the actual query host. This allows a third party to query on behalf on another host. This MUST be specified in the query.

- precision: This defines the precision of ALTO guidance required by the client. Current this supports three values: 0x01 for high, 0x02 for medium, 0x03 for low. ALTO server MAY choose to use more complex or less complex guidance algorithms on the backend for different precisions requested. The ALTO client MUST define a value for this field.

- metrics: This is a ranked list of constraints that denotes the interest of the client. Each metric constraint contains a metric, operator and value. The metricname is one of the standardized metric names supported by the specific ALTO service as defined in the ALTO configuration document. The operator field defines four values: 0x01 for less than, 0x02 for approximately equal to (within 10% of), 0x03 for greater than, 0x04 for NA. An operator of NA means that the querying host wants guidance on the metric but has no constraint on the metric value.

- destinations: This is a list of peers the querying host wishes to select from. The destinations are a list of host location attributes.

- oid: This is an optional field containing the object identifier the query host is aiming to transfer over the network. The querying host MAY choose to include this field and the ALTO server MAY choose to use it. This field allows ALTO services that manage object caches to include them in the guidance response by matching the oid. For example a querying host may only know 5 sources of an objects that are all interdomain sources of an object. If the oid is specified, the ALTO server could include network caches that match the query constraints in the list of peers returned to the querying host.

Guidance Query

- OPEN ISSUE: can we have a widely accepted method of generating the oid which makes referencing objects and identifying additional peers with the object easier. One method to use for generating the oid is content-based naming where the oid is the cryptographic hash of the object data.

Guidance Response

```
struct {
    HostLocationAttribute hla;
    MetricData metrics<0..2^8-1>;
   uint16 lifetime;
 } ResourceOwnerMetric;

struct {
ResourceOwnerMetric selection<0..2^32-1>
} Response;
```

Error response

```
public struct {
    uint16 error_code;
    opaque error_info<0..2^16-1>;
} ErrorResponse;
```

- Error_unauthorized
- Error_hla_not_found
- Error_overload
- Error_option_not_found
- Error_metric_not_found
- Error_no_matches

Example Use

- Use the ALTO query protocol with metric of latency less than 50ms as the first metric constraint and pDistance NA as the second query constraint. This query can be sent out to the ALTO service and a response received which will contain all peers ranked in terms of their latency followed by their pDistance.

- Node can make choice to first select all lower latency peers with a good pDistance value and choose remaining peers based on latency only