ECRIT H. Schulzrinne
Internet-Draft Columbia U.
Expires: December 20, 2005 June 18, 2005

Location-to-URL Mapping Protocol (LUMP)
draft-schulzrinne-ecrit-lump-00

Status of this Memo

This Internet-Draft will expire on December 20, 2005.

Copyright Notice

Abstract

LUMP (Location-to-URL Mapping Protocol) maps geographic locations, described as PIDF-LO objects containing civic or geospatial information, to one or more URLs. It is based on a standard RPC mechanism and supports updates. Clusters are used to ensure scaling and reliability. A flooding mechanism distributes top-level routing information. Naming authority can be delegated in any tree-like fashion, with multiple independent authorities for each level.

Table of Contents

1. Terminology 3
2. Definitions 3
3. Introduction 4
4. Introductory Example 5
5. Overview of System Operation 6
6. LUMP System Architecture 7
7. Resolver Discovery 11
8. Protocol Operations 12
  8.1 Query 12
    8.1.1 Query input 12
    8.1.2 Query Output 13
    8.1.3 Query Error 14
  8.2 Update 14
    8.2.1 Update Input 14
    8.2.2 Update Output 14
    8.2.3 Update Error 15
  8.3 Summary 15
9. Configuring Emergency Dial Strings 15
10. Security 16
11. References 17
  11.1 Normative References 17
  11.2 Informative References 17
    Author's Address 18
A. Acknowledgments 18
    Intellectual Property and Copyright Statements 19

1. Terminology

In this document, the key words "MUST", "MUSTNOT", "REQUIRED", "SHALL", "SHALLNOT", "SHOULD", "SHOULDNOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant implementations.

2. Definitions

In addition to the terms defined in [11], this document uses the following terms to describe LUMP:

authoritative resolver: Resolver that can provide the authoritative answer to a particular set of queries, e.g., covering a set of PIDF-LO civic labels or a particular region described by a geometric shape. In some (rare) cases of territorial disputes, two resolvers may be authoritative for the same region.

child: A child is a resolver that is authoritative for a subregion of a particular server. A child can in turn be parent.

cluster: A cluster is a group of resolver (servers) that all share the same mapping information and return the same results for queries. Clusters provide redundancy and share query load. Clusters are fully-meshed, i.e., they all exchange updates with each other.

complete: A civic mapping region is considered complete if it covers a set of hierarchical labels in its entirety, i.e., there is no other resolver that covers parts of the same region. (A complete mapping may have children that cover strict subsets of this region.) For example, a region spanning the whole country is complete, but a region spanning only some of the streets in a city is not.

hint: A hint provides a mapping from a region to a server name, used to short-cut mapping operations.

first resolver: The first resolver is the resolver contacted directly by the ESRP or end system to obtain a mapping. Architecturally, all resolvers can serve as first resolvers, although local policy may disallow this.

leaf: A resolver that has no children.

mapping: A mapping is a short-hand for 'mapping from a location object to one or more URLs describing either another mapping server or the desired PSAP URLs.

parent: A resolver that covers the region of all of its children. A resolver without a parent is a root resolver.

peer: A resolver maintains associations other resolvers, called peers. Peers synchronize their region maps.

querier: The resolver, ESRP or end system requesting a mapping.
region map: A data object describing a contiguous area covered by a resolver, either as a subset of a civic address or a geometric object.
root region map: A data object describing a contiguous area covered by a resolver, with no parent map.
resolver: The server providing (part of) the mapping service. Resolvers cooperate to offer the mapping service to queriers.
root resolver: A resolver without parents is a root resolver.

3. Introduction

The location-to-URL mapping protocol (LUMP) maps a civic or geospatial ocation, typically specified as a PIDF-LO object, to a set of URLs that describe the services available for that location. The initial application is the mapping of locations to the appropriate Public Safety Answering Point (PSAP) for emergency calling. It uses a common RPC protocol for its operations.

LUMP has the following properties, described more fully later in this document:

Satisfies the requirements [11] for mapping protocols.
LUMP supportes lookup as well as address validation for civic addresses.
LUMP re-uses of the most commonly used RPC protocol, SOAP, with a variety of transport and security options. (Other mechanisms, such as XML-RPC, may also work.) The choice is motivated by the availability of numerous well-tested implementations, both open and closed source, in just about any conceivable language framework (with the possible exception of Fortran and Cobol).
LUMP uses a robust clustering and replication architectures that distributes load as widely as possible, with every resolver as an entry point.
LUMP fully specifies mechanisms for distributing coverage-region information.
Mapping can be based on either civic or geospatial location information, with no performance penalty for either.
LUMP can be deployed bottom-deployment as well as top-down, with no need for a global coordinating body or the management of a global namespace or DNS name. The mechanism described does not require a country-level mapping server or a set of "root" servers.
Mapping services can be offered close to the access network, by the VSP/ASP, or by independent third parties.
LUMP supports a mechanism for updates and synchronization.
LUMP uses automated cluster replication with guaranteed convergence properties for maximum robustness [7].

LUMP supports split responsibility for a single civic hierarchy level. (Example: A city has three public safety agencies, with three PSAPs and independent mapping databases, each covering a subset of the streets in the city.)
LUMP can be extended to additional operations and data types.
Scalable both horizontally and vertically, i.e., any number of servers can support each subset of the mapping information and the number of levels is not bounded.
LUMP minimizes round trips by caching individual mappings as well as coverage regions ("hinting"). Unless otherwise desired, there is only one message exchange (roundtrip delay) between the ESRP or end system requesting a mapping and the designated resolver. This also facilitates reuse of TLS or other secure transport association across multiple queries.
LUMP supports both exact and approximate (best-guess) matching, controllable by the querier.
Mapping servers require only limited mutual trust.

LUMP combines aspects of directory lookup protocols such as IRIS [8] and hierarchical name mapping protocols such as DNS. However, it tries to avoid the constraints imposed by these earlier protocols designed for different applications. For example, it is not bound to having a resolver hierarchy that reflects the hierarchical nature of a civic location and does not have to try to fit the non-hierarchical nature of geospatial addresses into a label hierarchy. LUMP tries to avoid the notion of root servers and allows bottom-up deployment. LUMP supoprts updates, as this is necessary to design a robust replication system that allows LUMP nodes from different providers to become members of a cluster, without relying on unspecified protocols. Unlike DNS, secure channel associations are included in the design, as the fan-out at each level of the hierarchy is likely to be much lower. Also, LUMP is not encumbered by label and character set restrictions that make use of DNS cumbersome. Both exact and best-effort matches are possible.

4. Introductory Example

For this example, assume that there is a SIP-based VSPs V that offers a first resolver service to its customers. The VSP operates a cluster of such LUMP servers, advertised to their customers via DHCP. For simplicity, we only look at resolution by civic address; resolution by geo coordinates work exactly in the same fashion.

Assume that in the United States, each state operates a resolver, covering the counties or parishes in the state. In our example, there is no server covering all of the United States or larger regions. Each county in the state in turn has a list of coverage regions, typically consisting of one or more PSAPs. The state

servers have their own database that is not shared with the rest of country. Assume that the caller is located at 123 Broad Avenue, Bergen County, Leonia, New Jersey.

An end user affiliated with V1 needs to place an emergency call and dials "9-1-1". The end device translates this into an "sos" URI, which reaches the outbound proxy operated by V1, acting as an ESRP here. The ESRP issues a LUMP request to the local first resolver, RV1. RV1 has stored the coverage regions for all the states and matches the request to the New Jersey server, using the PIDF-LO location information contained in the SIP INVITE request for the lookup operation. Since it operates in recursive mode, it in turn queries the New Jersey server, say, lump:state.nj.example.gov. That server does not want to reveal more detailed information to the caller and simple returns a URL for the state-wide emergency services proxy, say sip:sos@emergency.nj.example.gov.

The ESRP routes the call to sip:sos@emergency.nj.example.gov, a SIP proxy server. In one or more resolution steps, that proxy server in turn consults a local LUMP server with the same PIDF-LO location information. Assume that the town of Leonia is served by two PSAPs, which do not share the same database. Streets south of a main road are served by one, those north by another. The state LUMP server only knows that Leonia has two such servers and issues a request to both, i.e., lump:north.leonianj.example.gov and lump: south.leonianj.example.gov. Broad Avenue is divided by this street, with 124 Broad Avenue happening to fall north of the dividing line. Both LUMP servers get the request and the northern server returns an answer, while the southern server indicates that this address is outside of its coverage region. The northern server returns the PSAP address, say, sip:police@leonianj.example.gov. The proxy simply routes the call to that location, including the location information.

This is only one of many possible deployment scenarios. As noted elsewhere, the area served by each server does not have to correspond to a particular civic address level or can span multiple levels. The referral graph can differ between civic and geospatial addresses and can utilize completely different servers, beyond the first resolver.

5. Overview of System Operation

A querier, such as an ESRP or end system, desiring to obtain a location mapping follows the steps below:

Identify a resolver: Using either DHCP [2], a service location protocol such as DNS-SD [9] or SLP [6], a using-protocol configuration protocol (e.g., [10] for SIP) or another configuration mechanism, the querier obtains one DNS name for a LUMP server cluster.

Determine a first resolver: The domain name obtained in the previous step is resolved using the associated SRV [3] resource record. The querier chooes the highest-priority server, and continues down the list if that server does not respond. As detailed in the SRV specification, a querier chooses randomly among multiple entries with the same weight. The use of DNSsec is RECOMMENDED.

Send query to first resolver: The querier sends a LUMP query to the resolver identified in the previous step, using an existing or newly-established secure transport association. The query contains a PIDF-LO [4] object. The resolver either determines that it is authoritative for the location contained in the query or it determines the root server for the location using region maps stored locally. In either case, the first resolver issues the same query provided by the initial querier to the appropriate resolver, which then recurses until it can determine a set of URLs for this location. The resolution path is recorded in the query result and returned to the initial querier as an ordered list of URL, priority tuples. If the query does not match any existing record, the query returns an appropriate error code. However, if the query allowed for approximate mapping, a URL may be returned, with an appropriate warning.

In the next section, we describe how LUMP works "behind the scenes" to perform this resolution.

6. LUMP System Architecture

A LUMP system consists of resolvers, organized into one or more clusters. Each cluster member provides the same information and offers load scaling and redundancy. Each cluster may be authoritative for a set of location-to-URL bindings or it may simply forward queries to other such clusters. Cluster members automatically synchronize their data stores with each other, so that updates made in any one cluster node propagate automatically to all other cluster nodes, even if some nodes were unavailable when the update was performed. Cluster nodes can (and should) be geographically distributed for increased failure tolerance. It is RECOMMENDED that each cluster contains at least two members. All cluster members are listed in a single DNS SRV record, typically, but not necessarily, with equal priority. Since all resolvers within a cluster offer equivalent services, we often use the terms resolver and resolver cluster interchangeably where the precise host identity does not matter.

Resolver clusters that are authoritative form a logical resolution hierarchy, i.e., resolvers can refer queries for more detailed resolution to other resolvers. The hierarchy is not tied to a particular element of the location object. For example, it does not have to follow a country, state/province, city, and street hierarchy. We refer to a resolver A referenced by by another resolver B as a child resolver in relation to that resolver B.

```
   /---------\            /-----------\
   | first   |    peer    | first     |
   | resolver|------------| resolver  |
   \----/----/            \-----\-----/
        |                        \
        |                         \
        |                          .
        |                          \
        |                           \
        |                 cluster    \
        |                 ............\.....
        |                 .  ,-,      ,',  .
  /-----\-----\           . |   ------   | .
  | first     |           . |   |    |   | .
  | resolver  |           .  `/'      `/'  .
  \-----------/           .   |        |   .
                          .  ,\,      ,\,  .
                          . |   ------   | .
                          . |   |    |   | .
                          .  `''      `''  .
                          ..................
```

In many cases, the degree of the tree will be modest. For example, if there were a resolver for the United States, it might have 51 child servers for the 50 states and the District of Columbia. We anticipate that fan-outs from 20 to 100 are common, as that seems to be a common span of control for each civic administrative level. For fan-outs of this order of magnitude, it becomes feasible for the parent resolver to maintain secure channel associations, e.g., via TLS, to all of its children, greatly accelerating the resolution process.

Thus, when receiving a query, each resolver checks if the query can be ansered locally. The answer may contain a pointer to another resolver. For example, a server for the state of New Jersey in the United States might contain the following database entries

```
   A1 A2         URL    resolver
US NJ Atlantic   -
US NJ Bergen     -
```

```
US NJ Monmouth   -
```

In this example, the resolver has local knowledge that it only needs to match country, A1 and A2 elements. All other PIDF-LO elements are ignored in selecting a matching entry. This example shows a non-leaf resolver that only points to other resolvers.

A leaf resolver contains at least some mapping URL, i.e., URLs of PSAPs. In the example below, Leonia is a town within Bergen county, with two streets, Broad and Grand.

```
US NJ Bergen Leonia Broad   sip:psap1@leonia.example.com
US NJ Bergen Leonia Grand   sip:psap2@leonia.example.com  xmpp:
```

Above, we assume that streets in Leonia are served by two different PSAPs, but contained in the same resolver.

A more complicated example is the case where PSAPs within a single city, for example, cannot agree to operate a single resolver, but rather have each PSAP operate its own for its own coverage area. The division might be by street names, sides of street, or even by service (fire vs. police.) The leaf servers have entries as above, but the server handling Leonia (e.g., at the county level) would contain an entry such as

```
US NJ Bergen Leonia lump:r1.example.com,
lump:r2.example.com    sip:psap@leonia.example.com
```

When a query for Grand Avenue, Leonia reaches this resolver, the resolver obtains two answers, r1.example.com and r2.example.com, from its database. Since it does not know which of two child servers for Leonia knows about the PSAP for Grand Avenue, it sends a query to both servers. Typically, one server will return a failure response, indicating that it does not contain such a mapping, while the other will respond with a PSAP URL. If both respond with a failure to resolve, the county server in this example would return a default PSAP URL, here sip:psap@leonia.example.com. In the hopefully unlikely case of dueling PSAPs that both are claiming to serve Grand Avenue, both would return an answer and the combined answer would be returned. This mechanism can also deal with the case that there is no single emergency contact, but that different emergency services maintain their own citizen-facing call center operations. In that case, both servers might return an answer, one indicating the URL for the fire service, another police service. (Alternatively, the query could constrain the service.)

While the example above returned two PSAP URLs, the same mechanism also works at non-leaf nodes to return resolver names. By explicitly

allowing for split authority, we avoid the notion of lame delegations.

(These examples do not imply that the database needs to be relational. An XML database, for example, might be used.)

As described in more detail later in this document, queries can ask to be treated recursively, i.e., where the resolver returns a final answer, or iteratively, where it returns a resolver name if it cannot provide a PSAP URL. (This is similar in spirit to the DNS approach.)

The description above shows how servers that are authoritative for a set of mappings obtain an answer, but does not solve the bootstrapping problem, namely finding the right first top-level server. This job is performed by so-called first resolvers, i.e., a set of resolvers that are directly contacted by queriers. In DNS terminology, LUMP makes all (first) resolvers "root" servers, i.e., capable of finding the right entry point into the tree. This not preclude operating LUMP in a manner similar to DNS, i.e., with a small number of root servers that handle the whole world, but we believe that coordination, deployment, robustness and administration are improved by allowing for a far more distributed entry point.

In LUMP, each first resolver must be equipped with a map for the top-level regions of the world, each served by a hierarchy of authoritative servers. There is no need for these areas to be contiguous or exclusive, i.e., it is possible for the same geographic spot to be claimed by two entities. The system works even if only small areas of the world participate initially, without having to agree on root servers. Thus, for example, we can defer the issue of an international coordination body well into the future. (Editorial aside: the difficulties in deploying ENUM illustrate that such coordination causes significant delays and overhead.)

As noted above, resolvers that are contacted directly by end systems or ESRPs are called first resolvers and all such first resolvers share a global region map, distributed by an application-layer broadcast mechanism. First resolvers may also be authoritative for a particular region, but that is not required. For example, a voice service provider might operate one or more resolvers that are used as first resolvers by its customers. Conversely, a resolver that is authoritative for a region may decide not to be able to serve as a first resolver and thus does not need to receive global region maps.

This mechanism does not scale indefinitely, but we believe that it readily supports thousands of top-level authoritative resolvers. This belief is based on the scaling properties of Usenet, which uses a vaguely similar architecture as the one proposed here.

The amount of data that needs to be distributed to all first resolvers is relatively small and likely to only see incremental updates as new regions are added or regions are split. Longer term, it appears likely that the number of such regions corresponds roughly to the number of countries, i.e., around 200. Regions described by civic addresses, e.g., a country or state, would have a single PIDF entry and a resolver URL. Regions described by a geospatial boundary would contain a GML polygon and a URL. It is hard to estimate bandwidth usage for distributing this information precisely, but reasonable estimates are probably measured in kilobytes per year.

First resolvers peer with other resolvers to exchange top-level LUMP request routing information. Each resolver can peer with as many other resolvers as it deems administratively appropriate, as long as the set of first resolver clusters form a connected graph. (It is sufficient, albeit unwise, that only one server in a cluster peers with other servers.)

If a new resolver covering a previously uncovered territory joins LUMP, it distributes an XMLDSIG-signed coverage map, consisting of a set of polygons to indicate geospatial coverage and/or a set of civic address labels and values to indicate civic coverage. These coverage regions are signed to prevent spoofing and to allow receiving resolvers to make policy choices if the same area is covered by two resolvers, e.g., for territories in dispute. (We assume that top-level regions are complete.)

When receiving a map from a peer, a resolver distributes a copy to each of its other peers, flooding the map to the whole graph.

When a new LUMP resolver joins a cluster or the overall LUMP graph, it requests the current set of regions from its peer. More precisely, it uses the XXX synchronization mechanism to determine whether it needs to update a peer. This avoids having multiply-connected peers receive multiple copies of the same region map. Somewhat simplified, a peer conveys to each peer a table of hash values reflecting the region maps it currently has stored. This mechanism also deals with memory loss in a resolver.

Like DNS zone files, coverage regions carry an identifier and timestamp to allow receivers to replace old regions with new regions. Region maps do not expire; they are valid until replaced. (Expiration is not necessary since new ones are pushed to all resolvers.)

7. Resolver Discovery

LUMP services may be operated by a variety of organizations and

entities, including Internet service providers, Internet access providers, voice service providers, and specialized LUMP service providers, such as public safety agencies or commercial database vendors. Each of these can either advertise their own servers or servers operated by other entities.

LUMP supports a range of resolver discovery mechanisms. Essentially, any discovery protocol may be used, including SLP [6], DNS-based [9] or UDDI. If the Internet service provider offers LUMP services, it may advertise these via DHCP. If the voice service provider offers LUMP services, it may include those in the SIP device configuration [10].

In general, it is advantageous to use a resolver that is close, in both a network topology and geographic sense, to the querier. Such proximity reduces the query latency due to reduced round-trip times and, in many cases, such servers will already have the necessary results cached, or at least pointers to appropriate authoritative resolvers and may already have established security associations with the appropriate resolver.

8. Protocol Operations

In this section, we describe the protocol operations. Detailed information about query and response parameter lists are described in WSDL in TBD.

8.1 Query

The query is the main operation in LUMP. The query includes a PIDF-LO object and returns a list of URLs, in addition to hints that can shorten the request path for future queries.

8.1.1 Query input

location object: The location object used for the query, typically as PIDF-LO.
location object format: The format of the location object, as an Internet media type (e.g., text/xml).
service: The service desired, e.g., "emergency.fire" or "emergency".
query precision: If set to "exact", the query fails if there is no precise match in all relevant location fields. If "partial", the matching algorithm may skip the least-significant parts of a civic address. If "soundex", the matching algorithm may use a sound-alike algorithm to find an approximate match. For example, the query "Main St" would match "Maine St". Other query precisions may be defined in the future. If the receiving resolver does not understand the query precision, it uses the "exact" matching

algorithm.

query mode: The query mode can be "recursive" or "iterative". In a recursive query, the resolver contacted for the query in turn attempts to resolve the query by contacting other servers if it is not authoritative for the location specified. In an iterative query, the resolver will return one or more LUMP URLs, in addition to any service URLs, that may be able to provide a more precise match. Resolvers MUST support an iterative query and SHOULD support a recursive query.

8.1.2 Query Output

URL list: The URL list enumerates all URLs discovered during the search. Each list element includes the URL, an indication of the service offered by the URL, a positive integer reflecting the priority (with zero having the highest priority), a match quality indicator (drawn from the values "exact", "partial", "soundex") and a host name indicating the resolver that provided this answer. If further searches are possible, a lump: URL is included. Normally, such lump: URLs are only included if the querier requested iterative resolution. The service indicator is optional and included if the URL only offers a subset of services, e.g., police or fire for emergency services. The match quality is included if not all parts of the civic address were used for matching. It indicates the lowest-granularity indication by its PIDF-LO element name, such as A6.

hint list: The hint list includes elements consisting of a LUMP URL, an expiration time and either a PIDF-LO containing civic information or a polygon. The hint indicates a region and its associated LUMP URL. For example, a hint with the region "CN=US A1= XXX" and URL=lump:nj.example.com would cause the resolver to direct all queries for this region to the nj.example.com server. The resolver MAY ignore hints. Hints are accumulated if a query is resolved recursively. To save space in responses, hints for geospatial regions may be subsets of the region covered. For example, instead of representing a country with a polygon having hundreds of line segments and precisely tracing the boundary, the hint may contain a simplified version that is strictly contained within the true boundary, but omits some regions close to the border. This causes most queries for that country to be resolved via the hinted URL, without having to store detailed maps.

location object: Optionally, the query MAY return a location object that add information missing from the query object. For example, where available, it may provide the geospatial location of a landmark specified as a civic address in the query.

path: A list of LUMP servers that was used for resolving the query, enumerated in the order used.

8.1.3 Query Error

reason code: Describes why the query failed, including server failure, no precise match, invalid data, refusal to recurse.
reason: Textual description of the error condition.
path: The list of servers that was used for resolution, with the last server in the list as the source of the error.

8.2 Update

The update operation is used to synchronize a server with a particular mapping from a PIDF-LO object to a set of URLs. This operation is used to inject new data into LUMP, by clusters to update other members of the cluster and to distribute region maps.

A receiver of an update behaves slightly differently depending on whether the update was received from an external entity (i.e., a node that is either a peer or a fellow cluster member) or from a peer or cluster member. If a resolver receives data from outside or a peer, it updates all fellow cluster members. If a node receives data from a peer or data that is marked as global from outside, it also updates all other peers. This floods all global data to all LUMP servers.

8.2.1 Update Input

location object type: A media type string indicating the type of the location object.
global: A flag that indicates whether this object is a top-level region description and thus to be flooded, or not. (As noted elsewhere, accidental flooding of non-top-level regions does no harm beyond wasting bandwidth between resolvers.)
region: The region, typically expressed as PIDF-LO or a polygon.
URL list: The list of URLs (services or LUMP) that are associated with that region.
replaces: Identifies, by hash value, the object that it replaces. This also allows a region to grow or shrink after an update.
expires: The time that the region-to-location mapping expires.

The location object, expiration time and URL are signed using XMLDSIG.

8.2.2 Update Output

TBD

8.2.3 Update Error

TBD

8.3 Summary

Resolvers within a cluster or peers exchange summary messages. A summary message contains a list of hashes that the sending node currently has within its object cache. The hashes include the same material covered by the XMLDSIG in the Update request above, i.e., include the expiration time. TBD: Bit vector instead?

If the recipient determines that the sender of the summary is missing a particular element, it sends the missing pieces using Update requests.

TBD: There are more efficient synchronization mechanisms, partially depending on the assumptions on the updates. See mSLP.

9. Configuring Emergency Dial Strings

For the foreseeable future, some user devices and software will emulate the user interface of a telephone, i.e., the only way to enter call address information is via a 12-button keypad. Also, emergency numbers are likely to used until essentially all communication devices feature IP connectivity and an alphanumeric keyboard. Unfortunately, more than 60 emergency numbers are in use throughout the world, with many of those numbers serving non-emergency purposes elsewhere, e.g., identifying repair or directory services. Countries also occasionally change their emergency numbers, for example, by selecting a number already in use in other countries of a region (such as 112 in Europe).

Thus, a system that allows devices to be used internationally to place emergency calls needs to allow devices to discover emergency numbers automatically. In the system proposed, these numbers are strictly of local significance and are generally not visible in call signaling messages.

For simplicity of presentation, this section assumes that emergency numbers are valid throughout a country, rather than, say, be restricted to a particular city. This appears likely to be true in countries likely to deploy IP-based emergency calling solutions. In addition, the solution proposed also works if certain countries do not use a national emergency number. There is no requirement that a country uses a single emergency number for all emergency services, such as fire, police, or rescue.

For the best user experience, systems should be able to discover two sets of numbers, namely those used in the user's home country and in the country the user is currently visiting. The user is most likely to remember the former, but a companion borrowing a device in an emergency may only know the local emergency numbers.

Determining home and local emergency numbers is a configuration problem, but unfortunately, existing configuration mechanisms are ill-suited for this purpose. For example, a DHCP server might be able to provide the local emergency number, but not the home numbers. Similarly, SIP configuration would be able to provide the numbers valid at the location of the SIP service provider, but even a SIP service provider with national footprint may serve customers that are visiting any number of other countries.

Since dial strings are represented as URLs [5], the problem of determining local and home emergency numbers is a problem of mapping locations to a set of URLs, i.e., exactly the problem that LUMP is solving already.

The mapping operation is almost exactly the same as for determining the emergency service URL. The only difference is that if a querier knows the civic location at least to the country level, it will use a query where the PIDF-LO only includes the country code. If it only knows its geospatial location, it has to include that longitude and latitude. The querier uses the service identifiers "dialstring.emergency", "dialstring.emergency.fire", etc. The resolver returns the appropriate set of URLs and, if a geospatial location was used in the query, the current region map for the country.

Within the LUMP system, emergency calling regions are global information, i.e., they are distributed using the peer broadcast mechanism described earlier. Thus, every resolver has access to all region mappings. This makes it possible that a querier can ask any resolver for this information, reducing the privacy threat of revealing its location outside of an emergency call. The privacy threat is further reduced by the long-lived nature of the information, i.e., in almost all cases, the querier will have already cached the national boundary information or country information on its first visit to the country, using the normal LUMP hinting mechanism. (Given the modest storage needs, a querier could even cache all boundary maps.)

10. Security

LUMP addresses the following security issues, usually through the underlying transport security associations:

Server impersonation: Queriers, cluster members and peers can assure themselves of the identity of the remote party by using the facilities in the underlying channel security mechanism, such as TLS.

Query or query result corruption: To avoid that an attacker can modify the query or its result, LUMP RECOMMENDS the use of channel security, such as TLS.

Region corruption: To avoid that a third party or an untrustworthy member of the LUMP server population introduces a region map that it is not authorized for, any peer introducing a new region map MUST sign the object by encapsulating the data into a CMS wrapper. A recipient MUST verify, through a local policy mechanism, that the signing entity is indeed authorized to speak for that region. Determining who can speak for a particular region is inherently difficult unless there is a small set of authorizing entities that resolvers can trust. Receiving resolvers should be particularly suspicious if an existing region map is replaced with a new one with a new resolver address.

Additional threats that need to be addressed by operational measures include denial-of-service attacks.

11. References

11.1 Normative References

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

[2] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[3] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[4] Peterson, J., "A Presence-based GEOPRIV Location Object Format", draft-ietf-geopriv-pidf-lo-03 (work in progress), September 2004.

[5] Rosen, B., "Dialstring parameter for the sip URI", draft-rosen-iptel-dialstring-01 (work in progress), February 2005.

11.2 Informative References

[6] Guttman, E., Perkins, C., Veizades, J., and M. Day, "Service Location Protocol, Version 2", RFC 2608, June 1999.

[7] Zhao, W., Schulzrinne, H., and E. Guttman, "Mesh-enhanced Service Location Protocol (mSLP)", RFC 3528, April 2003.

[8] Newton, A. and M. Sanz, "IRIS: The Internet Registry Information Service (IRIS) Core Protocol", RFC 3981, January 2005.

[9] Cheshire, S., "DNS-Based Service Discovery", draft-cheshire-dnsext-dns-sd-02 (work in progress), February 2004.

[10] Petrie, D., "A Framework for Session Initiation Protocol User Agent Profile Delivery", draft-ietf-sipping-config-framework-06 (work in progress), February 2005.

[11] Schulzrinne, H. and R. Marshall, "Requirements for Emergency Context Resolution with Internet Technologies", draft-schulzrinne-ecrit-requirements-00 (work in progress), May 2005.

Author's Address

Henning Schulzrinne
Columbia University
Department of Computer Science
450 Computer Science Building
New York, NY 10027
US

Phone: +1 212 939 7004
Email: hgs+ecrit@cs.columbia.edu
URI: http://www.cs.columbia.edu

Appendix A. Acknowledgments

provided helpful comments.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.